Flood and Traffic Wireless Monitoring System for Smart Cities

Thesis by

Moustafa Osama Ahmed Moussa

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

October, 2016

EXAMINATION COMMITTEE PAGE

The thesis of Moustafa Moussa is approved after the review of the examination committee.

Committee Chairperson: Christian Claudel
Committee Members: Mohamed-Slim Alouini, Xiangliang Zhang, Jeff Shamma, Mohammed Younes, Ranga Venkatsha Prasad

ABSTRACT

Flood and Traffic Wireless Monitoring System for Smart Cities

Moustafa Osama Ahmed Moussa

The convergence of computation, communication and sensing has led to the emergence of Wireless Sensor Networks (WSNs), which allow distributed monitoring of physical phenomena over extended areas. In this thesis, we focus on a dual flood and traffic flow WSN applicable to urban environments. This fixed sensing system is based on the combination of ultrasonic range-finding with remote temperature sensing, and can sense both phenomena with a high degree of accuracy. This enables the monitoring of urban areas to lessen the impact of catastrophic flood events, by monitoring flood parameters and traffic flow to enable public evacuation and early warning, allocate the resources efficiently or control the traffic to make cities more productive and smarter. We present an implementation of the device, and illustrate its performance in water level estimation and rain detection using a novel combination of L1 regularized reconstruction and machine learning algorithms on a 6-month dataset involving four different sensors. Our results show that water level can be estimated with an uncertainty of 1 cm using a combination of thermal sensing and ultrasonic distance measurements. The demonstration of the performance included the detection of an actual flash flood event using two sensors located in Umm Al Qura University (Mecca). Finally, we show that Lagrangian (mobile) sensors can be used to inexpensively increase the performance of the system with respect to traffic sensing. These sensors are based on Inertial Measurement Units (IMUs), which have never been investigated in the context of traffic flow monitoring before. We investigate the divergence of the speed estimation process, the lack of the calibration parameters of the system, and the problem of reconstructing vehicle trajectories evolving in a given transportation network. To address these problems, we propose an automatic calibration algorithm applicable to IMU-equipped ground vehicles, and an L1 regularized least squares formulation for vehicle speed estimation. Results show that this system can be used to generate accurate traffic monitoring data, and significantly outperforms GPS sensors (traditionally used as traffic flow sensors) in terms of cost, accuracy and reliability.

ACKNOWLEDGEMENTS

I would like to deliver my absolute gratitude to the people who helped make this thesis a reality after the blessings of Allah. Starting from my parents and my brothers (Osama Moussa, my father, Mouna Alrashidy, my mother, and my brothers Ahmed, Mohammed, and Yousef Moussa) for their support during my life in general and academic years specifically and I will be forever in their debts. My professor, Dr.Christian Claudel who paved the way for my learning process and made my journey not only enriching but exponentially fruitful and rewarding, along with Professor Mohammed Slim Alouini, who I owe so much, even before I joined KAUST. To my friends who without their support I would not have had the power nor strength to keep pushing forward in the hardest of days, Mohammed Shaqura, Ahmad Dehwah, Manal Andijani, Rayan Naser, Chahrazed Elmetnani, Mohammed Alfarhan, Mohammed Alsharif, Fadl Abdullatif, Ahmad Bahgat, Idris Ajia, Basma Othman, Walaa Fatheldine, Mohammad Zahana, Sally Ahmed, Amber Siddiqi, Rishabh Dutta, Huda Ibeid, Ikram Bukhdemi, Rasha Abdulhalim, Rana Alrabeh, Konpal Ali, Stephanie Saade, Ghaida Hadiydi, Mehdi Derishe, Ahmed Bahgat, Amir Zaher, Amir Nabil, Thamer Nouh, Ali Agha, Engy Khames, Ahmad Mabrouk, Mohammed Ghoniem, Asmaa Hasheesh, Abdullah Dehwa, Omar Dehwah, Abdulrahman Eljedaani, Mayada Alhashim, Hanin Alzubaidy, Ahmad Badri, Rewaa Jallal, and to everyone I have ever known during my spectacular journey, I will surely never forget you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Background and motivation

The development of communication, computation and sensing has led to the emergence of wireless sensor networks (WSNs), that allow distributed computation and monitoring over large areas. WSNs have the potential to be very effective tools to address and enhance a number of operational issues, including emergency response during catastrophic events [1], pollution monitoring [2, 3], or environmental monitoring [4]. Moreover, WSNs have the potential to offer economically appealing solutions to these problems (both in terms of hardware, software and deployment costs). WSNs have witnessed limited use in urban sensing applications to date, despite their significant benefits over traditional wired sensor network architectures such as [5]. While sensor networks were expected to have an explosive expansion in the 2000s, the actual number of deployed systems is substantially below the most pessimistic forecasts, as illustrated for instance by the `ON World, In-Stat, WTRS, Harbor Research` 2006 forecast which predicted that between 80 million to 150 million units would be shipped in 2010, while only 45 million units were shipped one year later in 2011. One of the biggest challenges emerging with WSNs is their relative lack of performance and reliability, which prevents the use of highly sophisticated processing software that would be required for many applications. In particular, small scale deployments in a controlled setting can be misleadingly simple, and do not show the real difficulties associated with extensive and long wireless sensor network deployments [6, 7, 8].

In urban environments, sensing systems [2, 9] include traffic flow sensing [10], smart parking [11], pollution sensing, or smart infrastructure applications, such as in [12, 13]. In this thesis, the sensor component of a prototype urban flood and traffic wireless monitoring system using Lagrangian and Eulerian sensing is investigated. We selected a solar powered, wireless sensing system to allow fast and economically efficient deployments, lowering the global cost of the installed system, because a large part of the overall cost of an urban wireless sensor network is due to installation. However, full wireless sensor networks present unique advantages over wired, externally powered (using the power grid for instance) sensor networks typically used for traffic sensing, such as the `PeMS` system [5], or the `Sensys` system [10], which has a hybrid architecture: it consists in a set wireless nodes connected to a base station, which is itself connected to internet through wires. While WSNs have been extensively used in the past, relatively few applications are related to urban sensing. In particular, WSN solutions in urban environments are rarely fully wireless, and usually consist in mixed architectures with a wired network of gateways and relatively small wireless networks around them. Climatic conditions can make successful sensor network deployments extremely difficult to achieve in the long term which arise in a large number of urban areas that are located in desert or arid environments, for example, in North America, central Asia or north Africa. In addition, in areas that are prone to rainfall or natural disasters, more challenges make the deployments and the value of accurate and reliable information increase significantly which is the essence of the development of WSNs.

Two thirds of the world's population live in cities, where 75% of these cities are actually costal. With the population expansions, energy requirements and climate change that results on many natural disasters, cities nowadays need to be offer safe and secure environments for its residents under any circumstances. Among these natural disasters, floods are the most commonly occurring [14], accounting for more than half of natural disasters worldwide. Floods caused more than 120,000 fatalities in the world between 1991 and 2005 [15], (resulting in over 6000 causalities annually) and are a major problem in many areas of the world. Recent examples of catastrophic flood events include the 2014 floods in Kashmir, India (between 300 and 400 casualties), the 2013 flash floods in Argentina (more than 75 casualties) or the 2014 floods in the regions of Mecca and Jeddah (Saudi Arabia) that resulted in more than 135 million USD of losses. Among floods, flash floods are short triggered events that usually peak in a couple of hours. Most flood fatalities are caused by flash floods, and most flash flood victims die due to drowning [16, 17], which could be avoided by providing realtime, accurate and efficient flash flood maps to the population (through standard information channels). Unfortunately, at the present time little warning exists beyond weather imaging and forecasts, which are nonspecific and unreliable, *i.e.* these warnings are associated with a very high false alarm rate. Monitoring floods in real time somehow requires sensing the flooding conditions [18], which includes water presence, water levels, water velocity and rain rate. Current available static water level sensors are only adjusted to river monitoring but not to situations in which the runoff does not correspond to permanent rivers. Instrumenting entire hydrological basins, which can cover hundreds of square miles, is economically infeasible, since the lifetime of a sensor network is usually less than the mean time between catastrophic flood events. Satellites are also unable to accurately monitor water levels and water flows remotely: optical measurements are impossible during floods due to the severe rains associated with floods, and the vertical resolution of currently available synthetic aperture radars is insufficient.

For all these reasons, a wireless sensor network would be the best approach to flash flood monitoring in cities. We discuss this in Chapter 4. To be economically viable, this flash flood sensor network must have a secondary sensing capability that justifies the cost of its purchase and deployment. Other critical issues in cities include for instance air pollution or traffic congestion monitoring. Intelligent Transportation Systems (ITS) are critical in reducing traffic congestion, increasing safety and productivity. Since traffic congestion is a worsening issue in most countries, monitoring congestion levels in real time is of critical importance: it not only enables better information to be relayed to users, but also paves the way to efficient traffic control system [19]. Traffic monitoring includes the measurement of vehicle speeds, flows, density and the classification of vehicles in different categories (for instance buses, trucks or light vehicles). Current ITS systems typically rely on wired sensor networks, for instance the PeMS system in California [20], which do not have specific power requirement constraints. Alternate systems do exist, such as wireless sensor networks [21], which are much easier and cheaper to install and operate.

Unlike existing traffic sensing infrastructures, the proposed traffic sensor network is to use both mobile (Lagrangian) traffic data (which is the objective of this thesis in this particular context) and fixed (Eulerian) , and to process the data at the node level (using a macroscopic traffic flow model) to generate traffic estimates. Static (Eulerian) traffic data is generated by fixed traffic flow sensors, while Lagrangian data is collected from vehicles equipped with a positioning device and a short-range transceiver described in Chapter 5. Given that this sensor network is to be deployed in an urban desert environment, we initially focused on all the operational constraints associated with the operation of a wireless sensor network in urban cities, including packaging, deployment limitations, data accuracy, energy management and long term reliability. The current state of the art in traffic flow estimation is provided using satellite systems that are rather unavailable in some urban areas because of the canyon effect, and even when GPS is available, it is often less accurate because of the multi-path effects. Therefore, we have investigated a satellite free solution based on inexpensive IMU devices that offers more benefits, particularly because of the context of detection that the GPS systems unable to achieve.

The first implementation of this system was done using `Libelium Waspmote`®, a commercial hardware platform based on `Arduino` platforms, which we believe is typical of the WSN products available on the market. Over a period of three years, we conducted several tests and deployments to evaluate the performance of the system, as well as its reliability (presented in Chapter 2). The experience obtained from these experiments has shaped the research activities of the thesis, which lie in hardware development, energy estimation, water level estimation, and traffic flow monitoring algorithms. These initial assessments and investigations of the systems led to the understanding of the lack of more capable microcontrollers that would offer better processing powers that would reduce the overall system cost and energy usage specifically dedicated to the applications of concern.

1.2 Related work

We currently focus on sensors that can monitor simultaneously flash floods and traffic congestion in urban environments. Multiple sensing principles can be thought of in the context of traffic monitoring, including magnetometers [22], RADAR (Radio Detection And Ranging), traffic cameras [23], acoustic sensors [24], [25], or LIDAR (Laser Infrared Detection And Ranging) [26]. These sensing methods are typically expensive, since they require advanced processing circuitry, specially in the case of cameras, RADARs and LIDARs. In contrast, passive infrared sensors (PIRs), [27], [28] and ultrasonic rangefinders [29] are low cost, reliable and small, and can be readily used for both sensing applications. They are particularly suitable for traffic monitoring systems as they can be easily deployed on the road sides or on traffic lights, and can measure without contacting the target. They can also monitor distances, which make them particularly suitable for the present flash flood monitoring application. In this thesis, we propose a new type of traffic and flash flood sensors based on the combination of ultrasonic rangefinders with one or multiple passive infrared temperature sensors. This sensor combination can be used as a backbone for a dual urban traffic/flash flood WSN, since it can monitor vehicle speeds, counts, density and vehicle types as well as pluviometry, water presence and water level with a relatively high degree of accuracy.

Current flash flood estimation algorithms rely on forward simulations of the associated hydrological basin using hydrological models, which are combined with real-time measurements (or prior information) using for example Extended Kalman Filtering [30] or Particle Filtering. The underlying models are typically the Shallow Water Equations (SWE), which are an approximation of the Navier Stokes equations for problems involving shallow water levels (which typically occur in floods). The shallow water equations are a partial differential equation (PDE) that describe the evolution of a body of water under the influence of gravity, ground friction, and rains. The state of the body of water is encoded by two functions, the velocity function and the water level function. In almost all instances of flash floods, the diffusive wave approximation to the shallow water equations (DSW, [31]) can be used. The DSW equations are a simplification of the shallow water equations, used to model flows where the vertical momentum is small relative to the horizontal. With this approximation a single function (for instance the level of water) is sufficient to describe the state of the system.

The main issue arising when dealing with hydrological models is their inaccuracy, particularly because of large uncertainties on the runoff coefficients or the terrain altitudes. As an example, parameters of the inundation models such as the Manning coefficient are highly uncertain, even with high resolution satellite maps or LIDAR surveys [32]. For instance, the Manning coefficient associated with boulder terrain can vary from 0.04 to 0.07, a 70 % relative uncertainty [33]. An example of river level estimation using the combination of hydrological models with rain rate data is illustrated in article [34] by Li et al, for monitoring and predicting floods in Africa. In this study [34], discharge rates of rivers are estimated using satellite precipitation data as well as ground precipitation data from a network of weather stations. Despite the very high accuracy of satellite precipitation data (with a spatial resolution on the order of a few hundred meters), the actual discharge rates in the rivers vary significantly from the results of the simulations. Furthermore, a relatively high proportion of floods occur in desert areas (for instance the 2014 Indian/Pakistan flash floods), and are not associated with rivers that exist for long periods of time, and for which no extensive satellite data allows the calibration of the model. Recent efforts have shifted towards adding real time flooding data (for instance in the form of water level or water velocity measurements) to the estimation process, to improve the quality of flood estimates over pure simulation. One such flood WSN is described in Basha et al [1], for monitoring and predicting water levels in an hydrological basin located in Nicaragua, using direct water-level measurements as well as precipitation data.

Other flood monitoring systems exist, in particular satellite-based or UAV-based (unmanned-air-vehicle) optical systems. For instance, an UAV-based system has been developed by [35]. More recently, in 2011, a Predator B drone from the US Customs and Border Protection office was used for flood mapping in Minnesota and North Dakota. However, in all of these operations, the focus of the operations was flood damage assessment, not real-time flood monitoring. All UAVs used in this context mapped the area using cameras, which cannot be applied to most flash flood sensing applications in which visibility is typically very low (high rains, mist, sand).

The use of synthetic aperture radars (SARs) in lieu of cameras is also discussed in [35]. While SARs are impervious to visibility problems, and can map extensive areas even during floods, their resolution is far too low for real-time monitoring [35], and are thus only used for post damage assessment. Typical SARs resolution is between 1 to 20m, though only 6 inches of fast moving water can knock an adult over [36]. An additional issue with SARs is their sensitivity to turbulence [37], which makes them impractical for flash floods, which are usually caused by thunderstorm cells.

1.3 Research Objectives

The main objectives of this thesis come here:

• To improve the energy optimization for the solar powered WSNs. A global optimization will be considered where the aim is to prolong the life time of the whole network and fairly utilize the resources. New algorithms will be applied as well as utilizing the available PDE models by tailoring that and investigating the possibility of introducing these models to WSNs in order to better optimize the energy.

• To accurately identify and forecast flash flood incidents through the estimation of water level and rain presence to provide the population early warnings and information on the impending flood (type, location, severity).

• To design a new system for probe sensing that offers several advantages over current, satellite-based systems. The system should be immune to multi-path effects and more economical than GPS-based systems and be able to be operational in very dense urban environments (with tunnels, underpasses and overpasses), and do not provide absolute location information, which greatly reduces the risk of privacy intrusion.

1.4 Main contributions

The main contributions of this thesis are as follows:

• Novel, dual use (traffic and flash flood) sensor system that addresses a key economic requirement in flash flood sensor networks (since flash floods happen very infrequently), providing high accuracy water level estimation (less than 1 cm) which enables flood detection and early warnings for cities to be more resilient and efficient.

• Providing a proven efficient and real time algorithms for traffic and flash flood monitoring application for better resource allocations and better decision making.

- Design of algorithms for generating traffic flow measurements from inertial data generated by probe traffic vehicles that provides better accuracy, and more economically efficient benefits.

Chapter 2

Solar-powered WSNs Architecture for traffic and flood sensing applications

In this chapter, the experimental and technical experience gained during the early years of the PhD research and tasks is discussed, that included the selection of the sensor network nodes, implementing wireless routing protocols, and development and deployment of several WSNs on the campus of KAUST. In order to study the feasibility of computation, communication and sensing capabilities of sensor network nodes, one need to consider many constraints related to planning, testing and environmental conditions that would make a successful long term deployments of WSNs, a real challenge for nowadays applications specifically, i.e. flood and traffic monitoring.

We had to survey the available wireless sensor networks architectures, and investigate, the design, previous deployments, energy management, the environmental perturbation, and visualization in order to comprehensively understand the hardware and software needs of such network scheme. All of this experience and investigations are in the following sections.

2.1 System design

2.1.1 Proposed urban flash flood and traffic sensing system architecture

We proposed a system network that consists in a different wireless sensors, connected to a database in a centralized manner. The vision was towards having a client queried database or feed others based on-demand location services, e.g. monitoring application that could extend beyond traffic or flash flood.

Sensor nodes (types, and roles):

The sensor nodes can be classified into three major roles: communication (between the WSN nodes), computation (distributed traffic inference, vehicle tracking, water level estimation, flood forecasting, fixed sensor data processing) and sensing (the actual sensing mechanism). Two types of sensing methods are available and considered in our study: Lagrangian (mobile) or Eulerian (fixed) sensing, resulting in Eulerian sensing nodes and Lagrangian sensing nodes. Eulerian sensing nodes consist in fixed traffic flow sensors, for instance inductive loop detectors [5], magnetometers or traffic cameras. The remaining nodes are called Lagrangian sensing nodes, and collect traffic data broadcasted by short range transponders onboard vehicles. All nodes are fixed in the urban environment to monitor, and are forming a wireless mesh network. The sink node (a node where all data are gathered and logs in the data to database) is placed in the range of the rest of the network. Also we are considering the presence of multiple databases/gateways which can reduce network load.

Principle of operation:

The operation of our system runs on the following principle. All the nodes (fixed and mobile) are contributing to form a wireless mesh network, which can be clustered potentially. Each set of nodes in a close proximity can form a subnetwork (cluster) for distributed computing purposes, that computes local sensing conditions independently from other clusters. In each subnetwork a local coordinator is chosen to supervise computations from the rest of the nodes in a certain subnetwork. This local coordinator node can be the node with the highest current CPU and/or energy level (depending on the limitation specified in the system)

Probe vehicles sensors (Lagrangian) broadcast their location and/or speed information to surrounding sensing nodes, which temporarily store this data as well as network connectivity data (RSSI, CRC). All location (if any), speed and connectivity data is forwarded to the local coordinator node. If no positional information is available to probe vehicles, the coordinator node estimates the corresponding vehicle positions in the road network using inputs from surrounding nodes. Vehicle mapping can be done through a variety of methods, for example using RSSI data, or map matching . The sensing nodes can also send some control parameters to the probe vehicle transceivers to adjust their transmission rate, to minimize packet collisions when a large number of users is present, or increase accuracy in the converse situation.

Also, along with the traffic data transmitted from the Lagrangian nodes, the coordinator receives data generated by the fixed sensor nodes in the subnetwork for the applications of concern, and this data can be categorized in two types:

1. Raw data measurements, e.g. point data of sensors, point velocity, flow or density data, generated in different areas of the road network and at different times.

2. Integrated (processed) values, such as the vehicle travel time between two given points, or corrected water level.

2.1.2 Technological choices

At the start of the assigned PhD work, we investigated and decided to select solar powered multi-hop wireless sensor networks over other possible technologies (for instance a network of 3G platforms connected to the cellular network, and powered by the grid) for multiple reasons:

- Reliability: the considered design is aiming at working 24/7 around the clock in order to provide cities with the correct and most accurate information. This is achieved through the independency of power grid or cellular networks, hence these two could be affected by the monitoring of a phenomena, such as the flood.

- Cost: the cost of each sensor node is currently of the same order of magnitude of the cost of deploying these nodes, even though no complex installation (connection to the grid, connection of data cables) is required. Connecting the system to the grid or using the 3G cellular network could solve a large number of problems, though these options make the system much more costly overall, which reflects a lot on the scalability of our sensor nodes.

- Privacy: the objective of our proposed system is to use distributed computing to estimate the state of traffic, for example, from anonymous short range user velocity data. To preserve privacy (even anonymous location tracks can be attacked in some circumstances, see for instance [38]), it is imperative to use local estimation and communication, which mandates the use of a multi-hop wireless sensor network.

- Redundancy: using the cellular network would prevent the system to operate when this network is saturated, which can happen during critical event (earthquakes, natural disasters...) during which traffic information is very useful to organize the evacuation of a city, resource allocation and the relief work.

2.2 Experimental experience

In this section, the experimental experience and setup used for the early PhD work is presented, including the hardware and the objectives of the deployments we had. These experience has led to many significant insights shaped the way for the rest of the dissertation work explained in the coming chapters.

2.2.1 System

For the first generation, we used `Libelium Waspmotes`®, a commercial platform based on the `Arduino` platforms, whose specifications are given in the following table.

Component	specifications
CPU	ATmega1281
Clock frequency	8 MHz
EEPROM	4 KB
RAM	8 KB
Flash	128 KB
SD	up to 2 GB
Radio	XBee-802.15.4-Pro
Frequency	2.4 GHz
TxPower	100 mW EIRP
Range	Up to 7000 m

Following the first experiments with this platform in 2011, some minor hardware and software modifications have been added. The platform used in this work only implements software reset, which does not clear the memory, and merely points the execution to the beginning of the program. Since memory leaks or memory corruption are a real possibility (as evidenced in earlier trials), we added some basic hardware reset circuitry (see section 2.3.1), and modified the API accordingly.

We also commented out large number of unused functions from the API to increase available memory.

2.2.2 Objectives of the WSN deployments

The general objective of the sensor network deployments was threefold.

- First and foremost, assessing the overall reliability of the platform, and the long

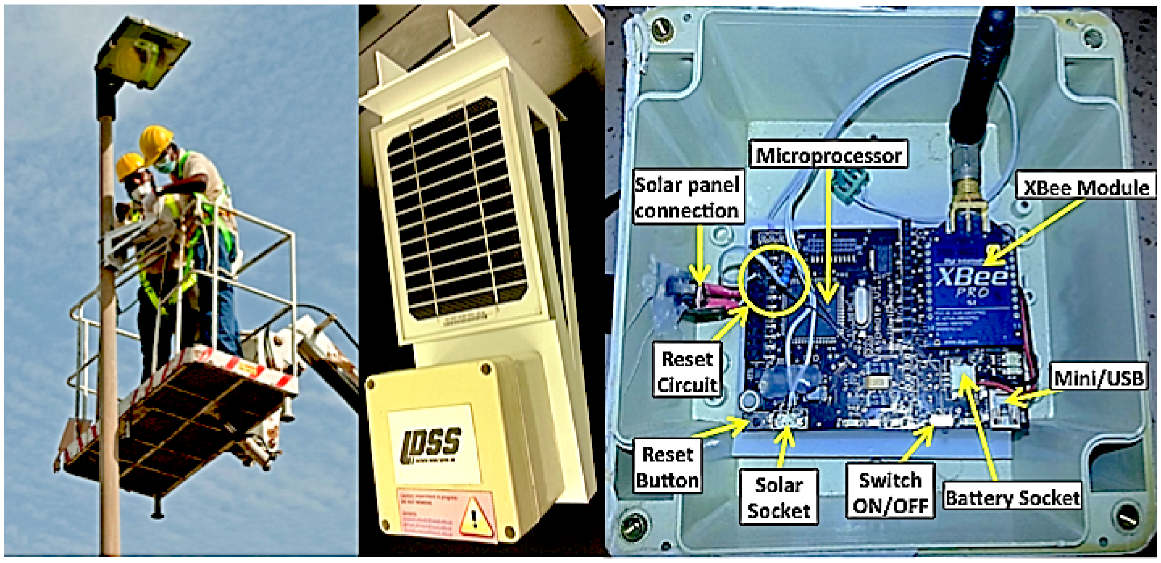


Figure 2.1: One assembled and deployed sensor node as part of the early experimental deplyoments.

term performance of the code.

- Second, to test the environmental impact of hot and desert conditions on the remainder of the hardware, including solar panels, batteries, enclosures and mounting frames. (more focus on this purpose is explained in section 2.4)

- Finally, the data generated by the sensor network (in particular the energy data) is invaluable to determine what the proper cost to energy dependence should be. We are using this data on sensor network simulators to optimize the cost function parameters.

2.2.3 Major Deployments (KAUST Campus)

There have been many experiments that were conducted to evaluate and test the network over the period of two years (2011- 2013):

- The first deployment was carried out between November 2011 to March 2012, and consisted in 12 motes plus a sink. The motes were running a simple protocol in which the routing tables were fixed, and the radius of the network was only two hops. The nodes generated energy and RSSI data for all incoming links. We used two different transmitter modules: `XBee-802.15.4` and `XBee-802.15.4-Pro` (with higher power) to confirm that they did not differ significantly (beyond their radiated power levels). The routing protocol was based on unicasting energy and RSSI messages to fixed successors and by broadcasting test messages to allow RSSI updates. The deployment worked well because of the relative simplicity of the code, the simplicity of the topology and the lack of reliance on SD cards for storing network route. However such a networking scheme is very limited: it only allows fixed topologies, and requires each node to operate a code with different parameters (fixed successors).

- The second deployment was carried out in May 2012, and consisted in a total of 80 nodes deployed (though no more than 52 were operational at any given time). The maximal radius of the network was 7 hops. The network was tested from 18th of May to 9th of June 2012 to assess the protocol functionality besides testing the energy, RSSI and connectivity of the motes and their links . All transceivers used in this deployment were `XBee Pro`. Node cost, node energy and link RSSI (with the associated number of packets received during a time cycle) were sent to the sink every cycle (total cycle duration was 5 minutes, though it is set by the sink during the discovery phase and can be modified). Link failure detection and propagation was implemented, and has been confirmed to work on at least two instances (see Figure 2.2 for one of such instances).

  It should be noted that two weeks were required for the deployment of this large sensor network to be completed, including the maintenance of some of these nodes. Figure 2.2 shows our web visualizer during a time at which the sensor network was near its performance peak. The number of responding nodes sharply dropped thereafter, particularly after a network discovery during which the operating parameters significantly deviated from the tested values. The sensor network became largely unresponsive by July 2012. Overall, the experiment had mixed outcomes: success on the short term, failure on the long term, because of the lack of robustness in the code and of our overly optimistic assumptions.

- Based on additional unit tests following the previous experiment, we discovered that the SD cards were causing irremediable problems. We thus developed a newer, more robust (albeit less energy efficient) version of the (code)/node software in which no SD card is used, and in which node cost is updated periodically by nodes based on messages from adjacent nodes. Because of the difficulty and cost of deployment, we chose this time to place 40 nodes on low-level obstacles, near the campus area. Operations started in December 2012, and are still ongoing as of April 2013. In this deployment, range has been severely reduced because of Fresnel zone obstruction by the obstacles and by the ground.

2.3 Lessons learned from the WSN deployments

We have gathered significant field experience following the multiple deployments carried out during the first three years of my PhD work along with lab team at KAUST. These lessons learned helped determine the research priorities of the group and my own thesis work, by underlining the most critical issues of the current sensing platform (which we believe is typical of commercially available systems). Some of the issues we found have implications for all testing environments, while some are specific to the urban and desert nature of the test location. The city in which the tests were carried out has a typical temperature ranging between 25 C (low) to 45 C (high) during the tested time frames, and is tens of square kilometers wide.

The main lessons learned are summarized in the list below:

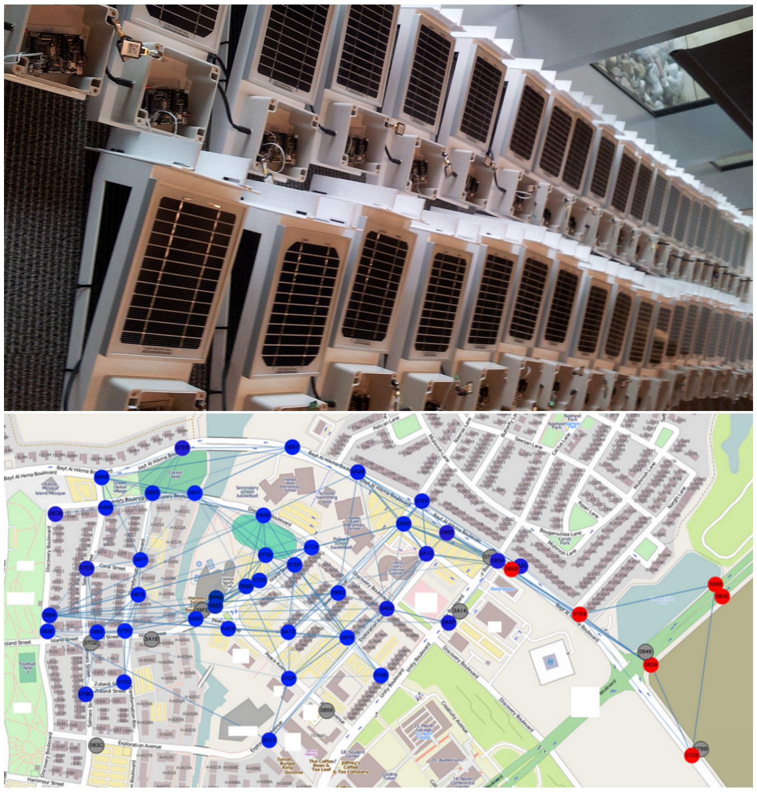


Figure 2.2: Our 2nd major deployment of 80 motes in May-June 2012, on the campus of KAUST (max latency: 1 hour/12 cycles)

1. Software and hardware faults are the regular issue to expect, in particular for mid term and long term deployments (many days or months). Debugging is always a complex operation, and a large number of bugs come from the API (the software libraries of the wireless motes/nodes).

2. A successful WSN deployment cannot be called so until it operates on the required period of time. As an example, our second experiment (80 nodes deployed among which 52 were functional) was successful during a few days, before failing catastrophically over the next few weeks.

3. Hardware specifications and data sheets are overly hopeful. Advertised ranges or data rates are very far from what can be achieved. For example, the data rate of an `XBee` transceiver is 256 kbps in reception, while in practice only about 3-4 packets per second (3.2 kbps) can be achieved due to API limitations.

4. Current commercially available systems have good performance when operating at extremely low duty cycles (for instance parking, or smart building applications). However, they are unsuitable (in the long term) to applications for which the duty cycle of sensing/computations is high, and to applications for which multiple sensing/computing applications exist.

5. Issues that might seem minor, such as, code structure, packaging, visualization and database take a substantially high amount of time

6. Results of previous deployment experiences (in the literature) in vastly different environmental conditions are not transposable to our experiment given all the unique conditions. While a number of system issues have been experienced by numerous groups, specific issues caused by the climate, the environmental conditions associated with the operation of solar panels and by the high-power high-range (smart-city) nature of the deployment.

We now describe the challenges we faced, emphasizing on hardware performance, API limitations, and limitations caused by the platform design. All these considerations were integrated by our team to design a more advanced platform, similar in philosophy to the IMote2 [39] and based on a `ARM Cortex M4` processor, which is used in the following chapter and explained in greater details which I helped develop.

2.3.1 Design challenges

Solar panels performance:

One might presume that the availability of sunlight in a desert environment such as ours with a large solar panel that can leverage this enormous energy would be enough to operate a WSN. Indeed, by that time our nodes were powered by a 3 W solar panels, which is considerably higher than the average power draw (90 mW for a 40% duty cycle). Actually, only one hour of direct solar panel exposure is enough to power our motes for an entire day, theoretically. However, in practice, a high number of nodes ends up being out of energy, due to two main factors, i.e. environmental conditions and shadow patterns.

The first factor is environmental conditions such as dust and humidity: humidity is high whenever the desert urban area is located near a water body. Dust storms deposit a layer of dust, which can cause more than 90 % loss in solar panel output. Humidity can turn the layer of dust into a mud, preventing it from being blown away by high winds. To complicate matters, dust deposition on nodes is far from being uniform, and is not only a function of solar panel orientation, but also depends upon its position in the city as well (since winds vary significantly in a city), as evidenced by Figure 2.3. While solar panels were all distributed within a 1 sqaure km area, dust accumulation on the solar panels varies a lot, which makes it hard to infer solar energy availability and mandates energy-aware protocols.

The second factor is the unpredictable shadow patterns caused by surrounding

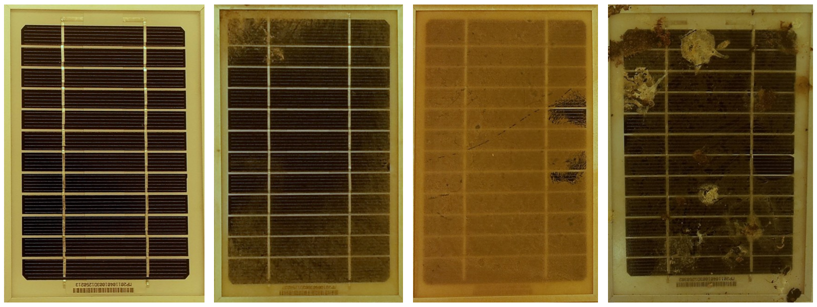


Figure 2.3: Irregular dust and debris accumulation on solar panels after a two month experiment. Left: solar panel before deployment. Center left: low dust accumulation. Center right: high dust accumulation (fingerprints visible on the right are caused by node handling after removal). Right: solar panel littered with debris from trees and birds.

buildings (including the street light itself) in cities. While we always deploy nodes by orienting them towards the sun, shadows vary significantly during the year. In particular, a successful deployment in summer might very well fail during winter time as cast shadows are much more prominent. Cast shadows can also reduce solar panel output by 90 %: we measured 20 mA at 12 V for a 3 W peak solar panel in a shadow.

Consequently, this will have a direct effect on the energy levels of the network motes, as illustrated in Figure 2.4. The energy patterns of the motes can differ in a significant way, even for similar node orientations. The main causes of node energy variability are:

- Motes positions on the maps are different, which means very different shadow patterns.
- The orientation of the steel frame (which is sometimes dictated by connectivity constraints and not energy constraints) can dramatically affect energy level of motes (the power output).
- Debris and dust, which accumulate at different rates.
- Battery capacities are a function of the condition of the battery, which depends upon its temperature and charge characteristics [40] (also explained in

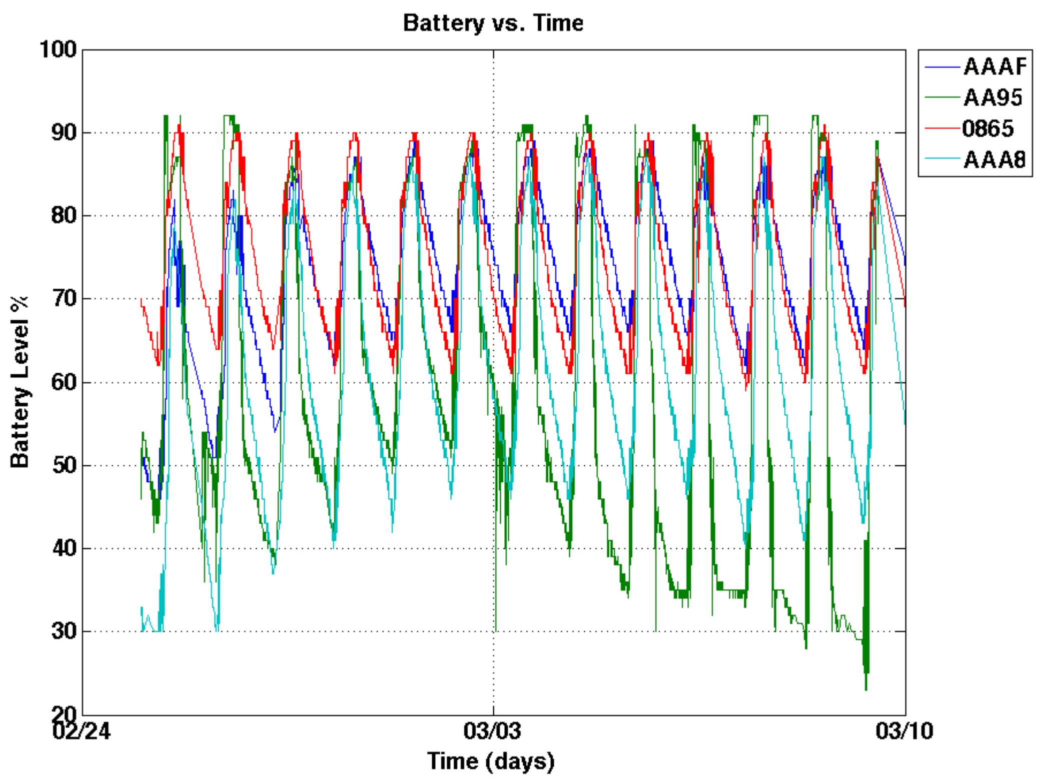


Figure 2.4: Battery energy evolution during 14 days for four different motes

Chapter 3). Figure 2.4 illustrates some dramatic variations of battery capacity: while all batteries are charged up to a similar maximal value, node AA95 has a fairly fast discharge rate during nights, though its energy consumption does not significantly differ from other nodes.

Packaging:

Packaging and enclosures are essential requirement for outdoors operation. Its design is a function of a number of factors, including material cost, weight, RF transparency and robustness. Since nodes were deployed high on street lights to enhance connectivity (see Figure 2.1), the location of the antenna (transceiver) relative to the street light column is important. From a structural engineering prospective, it is easier to place the mote, battery and enclosure (which concentrates most of the weight) near the street column, to reduce both weight and torque caused by wind drag. However, from a communication standpoint, this causes a significant loss of range due to the

Fresnel zone obstruction by the metallic street light column. The first deployment in November 2011 showed indeed that the RF signal attenuation by the street light in this configuration was unacceptably high, and the design was thereby updated (Figure 2.1). The enclosure is made of plastic (with some silicone sealant) and does not significantly attenuate the signal. To mount the solar panel and the enclosure, we used a steel frame, which was easy to prototype, albeit heavy and corrosion prone (despite being painted with a heat treated paint). For the next generations, we designed a new frame in Aluminum, with better weight and corrosion resistance characteristics. In addition, as initially designed, opening an enclosure required removing the silicone sealant and unscrewing the cover. The following design contains an external switch, an external USB socket as well as six external programmable LEDs. LEDs are very important to quickly check in the field that nodes are powered on and that solar panels are connected to the circuit board. This newer proposed design, illustrated in Figure 2.5, also allows the tilting of the solar panel while maintaining a vertical orientation for the node antenna and is considered in later chapters in this thesis. This was not the case with the previous system (explained in this chapter), negatively impacting its performance.

Hardware interface and bus sharing:
Bus sharing could be an efficient way of utilizing and optimizing the size of hardware device, from a hardware designer prospective. When planning for that, it is important to classify the tasks or chips connected to satisfy the user convenient as well as extending the life time of the device. In the commercial system used for at that time, UARTs (Universal Asynchronous Receiver/Transmitter) have been shared with many integrated circuits (ICs). Out of several ICs, the `XBee` module and the mini USB connector are using the same UART, which requires the user to unplug the `XBee` when uploading the code through the USB connector. However, this decision to share the same UART brings much more than a small inconvenience in practice. It gives a

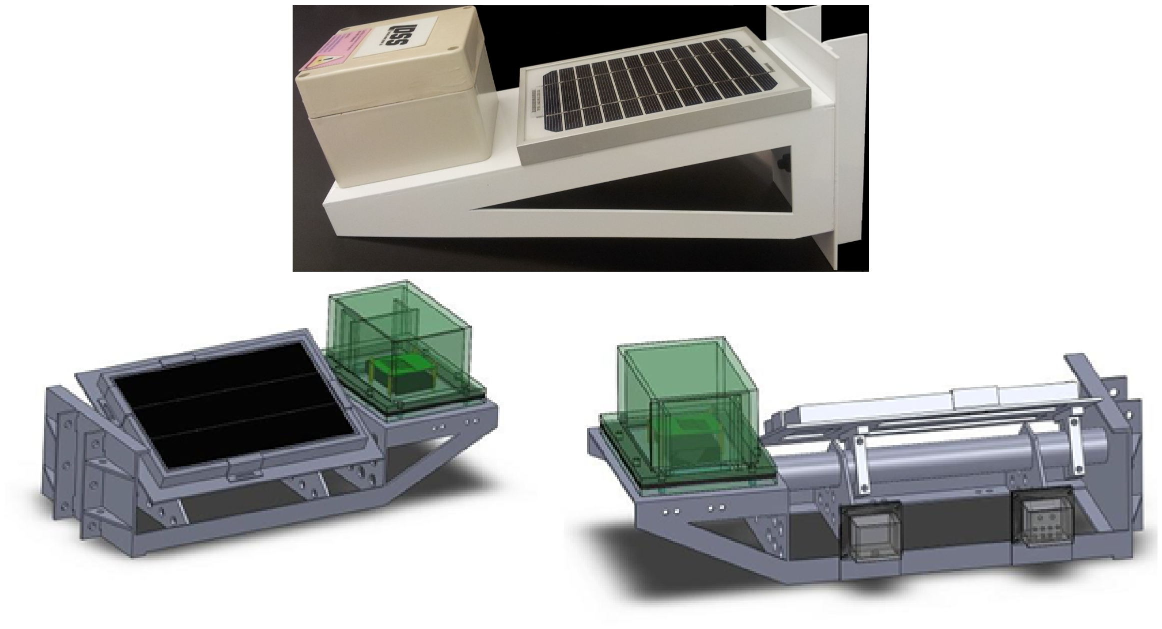


Figure 2.5: Top: node packaging for the first generation deployments explained in this chapter. Bottom: updated packaging (CAD design) explained in Chapter 4.

higher chance for dust or humidity to enter the sockets when code uploading is done in the field. It also prevents the use of an external USB socket on the enclosure, thereby forcing the opening of the system each time a code upgrade is necessary, dramatically affecting the overall reliability. The new platform considered in the following chapters solves this issue.

Reset circuitry:

Out-of-date software degrades the performance of the WSN due to corrupted data, unbounded resource consumption and the accumulation of numerical errors [41]. No matter how thoroughly the code is tested, it is always possible to have a memory leak caused by some part of software. We found that resetting the mote periodically is a solution to this problem. It can be done either virtually (based on software), or physically (hard reset). In the platform used at that moment, the node reset function merely points the execution to the beginning of the code (software reset), and does not clear the memory. Thus, this function is not very practical: a node that is hanging will not self-recover. Node crashes happen very often in practice due to various API bugs or hardware limitations. To allow hardware reset, a simple physical reset circuit (a resistor connected to a GPIO and soldered to the reset pin of the microcontroller) was implemented. Consequently, the protocol was designed to support new motes rejoining the network and synchronizing with surrounding nodes at any time.

The new platform that I helped develop with the rest of the research group contains a self-reset circuitry connected to an independent RTC, which will reset the mote even if the microcontroller or transceiver are unresponsive.

2.3.2 Hardware limitations and their impact on experiments

Memory

To build an advanced sensing system that can perform complex algorithms (such as machine learning, and smart preprocessing), memory availability is an important requirement to consider. Thus, a limitation on the memory brings severe restrictions on what can be implemented in practice. The memory specifications used in the early deployment platforms are as follows:

- Static random-access memory (SRAM): the hardware platform used initially has 8 kB as SRAM, with only 2 kB free. Handling an incoming packet requires 0.7 kB. Commenting some unused libraries brought back an additional 2 kB, though free memory is still insufficient to implement anything but the simplest algorithms or functions. To complicate matters, the memory management does not handle memory fragmentation well. In particular, the *Freememory* function that is used to check free SRAM in the API is not handling pointers correctly whenever memory is fragmented, and returns a lower value than the true available SRAM. We validated this by allocating and freeing some memory, with seemingly permanent memory loss. Though the microcontroller senses the true available memory at all times and can allocate it, not being able to track memory usage is very problematic in practice, in particular since free SRAM is very low.

- Secure Digital (SD) card: the `Waspmote` supports up to 2 GB (FAT 16) SD cards. While this provides extensive storage, it has its own drawbacks. One of the main issues with the SD card lies in its slow writing and access times. Lab tests showed that writing one byte to the SD card takes 23 ms, while reading takes 12 ms. In addition to the hardware speed, the SD card library caused random mote resets when accessing the SD card. These bugs, which caused the progressive failure of the sensor network deployed in May 2012, were largely solved in the latest API version released by `Libelium` early 2013.

- Flash ROM : this platform has a programmable flash memory of 128 kB. Even though this number seems respectable, we were close to reaching its limits with the routing protocol alone. Indeed, our routing protocol was occupying more than 102 kB (for a total length of more than 6000 lines).

All the above mentioned constrained have an influence on what can or cannot be done in practice. For example, to determine the route costs, one need to access the SD card for routes, for RSSI values and for energy statuses data. This is because the SRAM has a limited space and loading all of these values is impossible. Therefore, we have to calculate the cost of each route, individually. Each cost computation requires the loading of a route, its associated RSSIs values, and the associated node energy statuses. As mentioned earlier, reading the SD card takes approximately 12 ms. For a sensor network of 80 nodes, a few hundred links, computing one cost would take on the order of a second. This would have to be multiplied by the number of routes, which is typically large, in particular for nodes at higher levels where it reached hundreds. This is the reason for which we stored the link RSSI data with the routes during the discovery phase, allowing all data to be loaded together. However the downside of this is that link RSSIs values cannot be updated, as this would require reading and overwriting the RSSI values in a large number of places in the SD card, which is again too slow to be implemented. This shows how some seemingly inconsequential hardware limitations severely impact the overall system capabilities.

We avoided the use of an SD card altogether in our third deployment. This made the system much more robust with more than 30 days of successful deployment and allowed some extra free SRAM to be available (since SD libraries were not called nor required). We were not able construct advanced routing, scheduling or sensing algorithms, because of the extremely low available memory (both ROM and SRAM). Though some work could have been done at the API level to improve both access and writing times, as well as reliability, the SD card is not, with no doubt, a valid solution to the lack of memory at that point. These considerations together with the project requirements motivated the design and development of a sensing platform explained in Chapter 4. It contains 192 kB of SRAM (128 kB in practice) and 1 MB of ROM (768 kB in practice), an external flash memory (32 MB) as well as an SD card slot. This should allow more advanced computations to take place, though memory is still highly constrained compared to an embedded computer.

Connectors, switches and battery faults:
Short and long term deployment reliability is affected significantly by the connectors, switches and batteries used during these deployments. Switch faults can both cause a mote to fail to switch on as commanded, or to operate even with the switch placed in off mode (which occurred in one of our tests). In addition to switches, connectors are notoriously unreliable, even if when they are professionally soldered. Vibrations occurring during transport on flatbed trucks, or during deployment can break the relatively fragile leads used to connect the board to the solar panel. One of the most common problems that we faced is faulty solar panel connectors, and faulty jumpers on hardware platforms (though the company `Libelium` provided us jumper-free `Waspmotes` following our requirement. The newest `Waspmotes` versions are also

jumper-free). During our largest outdoors deployment, we collected all nodes at a given location for post deployment analysis, which showed the following problems:

- 26% of the motes were completely out of battery due to unreliable solar panel connection (bad contact).
- 24% of the motes have unreliable switches.
- 50% of the motes only had an “on” LED functioning properly (i.e. on when the mote is powered on).

These above numbers and facts are high, and are probably related to the numerous pre tests conducted with open enclosures (to facilitate code upload without always having to screw and seal the upper cover).

The `Waspmotes` are supplied by default with a 2.3 A.h Li-ion battery. While the capacity of this battery theoretically allows more than two days in listening mode, battery capacity considerably degraded in the hotter conditions associated with the urban, desert environment (discussed in more details in Chapter 3). While Li-ion batteries have a very good energy density, and a sufficient power density, their performance degrades significantly with temperature. We have estimated their maximal operating temperature to 45-50 °C, which is a common figure for Li-ion batteries [42]. The ambient temperature at the deployment site was always below 45 °C, though the box temperature occasionally reached 50 °C in the sun (measured with a remote IR temperature sensor). In these conditions, the degradation in battery performance was very fast, particularly for batteries that had a high charge. Figure 2.6 shows one of the batteries that was recovered following the deployment of May-June 2012. This battery was dramatically affected by the outdoor temperature compared to a normal battery, and suffered from “battery bloat”, in which the outer shell ruptured, causing a dramatic loss in capacity. While the battery shown in Figure 2.6 represents an extreme case, most batteries were affected at various degrees by this phenomenon.

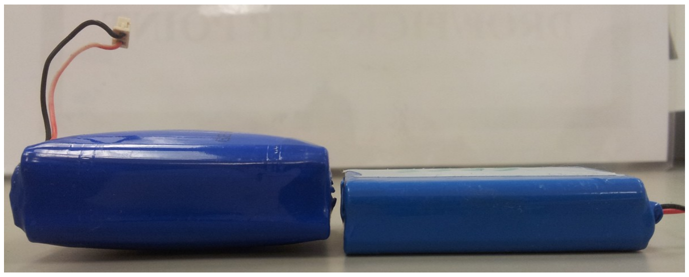


Figure 2.6: Left: effects of high outdoor temperature and high charge on a Li-ion battery. Right: A new Li-ion battery

For the coming explained next generation of sensor nodes, Lithium iron phosphate ($LiFePO4$) batteries will be used as a primary rechargeable battery. Though their energy density is slightly lower, their maximal operating temperature is much higher than Li-ion batteries, and have a much higher maximal number of charge/discharge cycles. The battery capacity was extended to 8 Ah, though it will probably be extended further to guarantee high enough reliability during actual operations.

Transceivers and their limitations (the wireless modules):

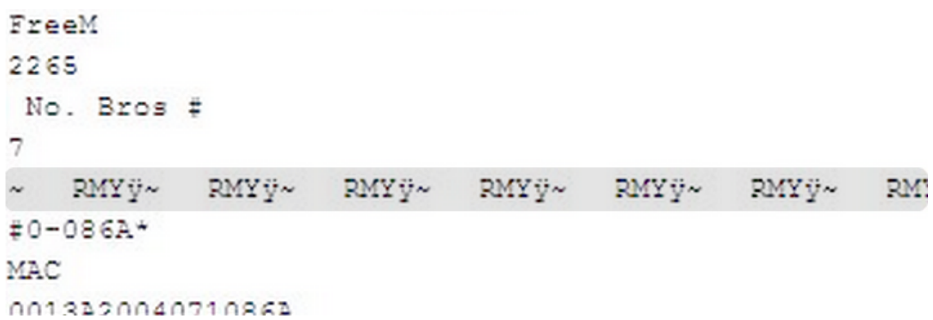
The wireless modules used in all platforms for the second and third deployments in the early work were `XBee-Pro` modules implementing the IEEE 802.15.4 standard at 2.4 GHz. We set the modules to their maximum power (i.e. 63 mW), which results on 100 mW EIRP, that is legal in most countries. Power consumption is relatively high with a transmission current of 180 mA at this emission level. While reception requires 50 mA, making the wireless transceivers the main energy consumption entity in this hardware platform (which is in most of the WSN deployment cases)

Since the firmware of the `XBee` is a closed source, we cannot program its protocol which is perhaps the best approach. Therefore, our protocol have to be implemented at the micrcontroller level. The `Waspmote` API has some useful functions to set up a wireless network, which allows programming at a higher level. We can also send commands directly to the `XBee` transceiver by setting it to a transparent mode. Also, the texttt Waspmote API does not support selective mechanism for clearing packets, instead it clears the whole buffer (including the non-corrupted messages) which is also considered as major limitation using these kind of transceivers.

In fact, the most serious issues occur on transmission and reception rates, as well as link dissymmetry issues. The `XBee` transceiver specifications tell that it is capable of 256 kbps transmission rate. However, we found that the actual maximal transmission rate is much lower. First, the API only supports serial data rates up to 38400 bauds. Higher rates are possible, though they lead to a much higher number of transmission errors. This gives a theoretical maximum of 4800 bytes per second, or 41 packets of 115 bytes per second. However, unicasting or broadcasting a 115 byte (including overhead) packet with the `Waspmote` API takes about 190 ms, which limits the transmission rate to less than 500 bytes per second (excluding packet overhead). Of course, setting the `XBee` in transparent mode leads to higher value, but remove support for important features such as encryption.

Though the actual transmission rate that we found is quite low, it does not cause much problems besides a smaller bandwidth. A greater anxiety arises when trying to send data through the `XBee` while the latter is handling received data. In this case, the mote can crash temporarily (for durations up to 30 s) while printing a packet header before recovering and eventually transmitting the packet, as illustrated in Figure 2.7. During this time, the microcontroller tries to communicate with the `XBee`, but fails. This error message can easily be reproduced by unplugging the transceiver from the mote while in operation. These problems are probably caused by the lack of optimization of the API used to operate the `XBee` module. This significantly limit the maximum data rate at which the nodes can operate, and can lead to a random data loss.

We found out during our debugging of the deployment in May-June 2012 that the microcontroller can crash while reading data from the transceiver, waiting for



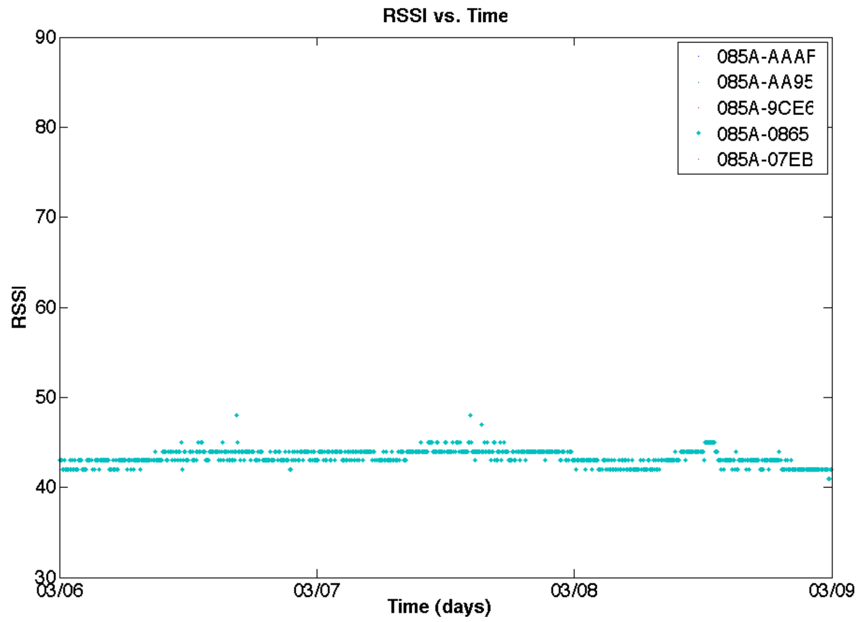
```
FreeM
2265
 No. Bros #
7
~   RMYÿ~   RMYÿ~   RMYÿ~   RMYÿ~   RMYÿ~   RMYÿ~   RM
#0-086A*
MAC
0013A2004071086A
```

Figure 2.7: Representation of pre-packets sent by the microcontroller (gray) before the packet ”#0-086A*” is eventually sent successfully. This happens when the microcontroller mote is unable to successfully send a packet to the transceiver.

a termination byte that never appears (the reading function implemented in the API did not have an execution timeout). After more than one year of operation, we discovered that the `XBee` transceivers can have much shorter range than expected, leading to considerably inconsistent links properties with surrounding nodes. Though dissymmetrical links sometimes happen, with up to 10% packet loss dissymmetry reported in [43], Figure 2.8 illustrates an extreme example, in which a mote ”085A” has five incoming links but only one outgoing link. This example obtained from data generated in March 2013, during the third deployment (initiated in December 2012). Explaining these results is complex: the cause of this dissymmetry is not solely related to outgoing signal strength, since the links $085A \rightarrow 0865$ and $0865 \rightarrow 085A$ have a nearly identical RSSI. One possible explanation could be a fault in the transmitter of 085A that affect the emitted signal.

2.3.3 Analoge to Digital Converters (ADC) faults

ADCs usage is essential in all analog sensing operations and as any other component, they can fail and it could lead to important consequences. In Figure 2.9 the failure of an ADC used to measure battery voltage is illustrated. Not only incorrect readings is resulted this way, but also pushes the mote to enter a sleep mode, which can be seen from this figure the sparsity in the readings. It also affects the cost function



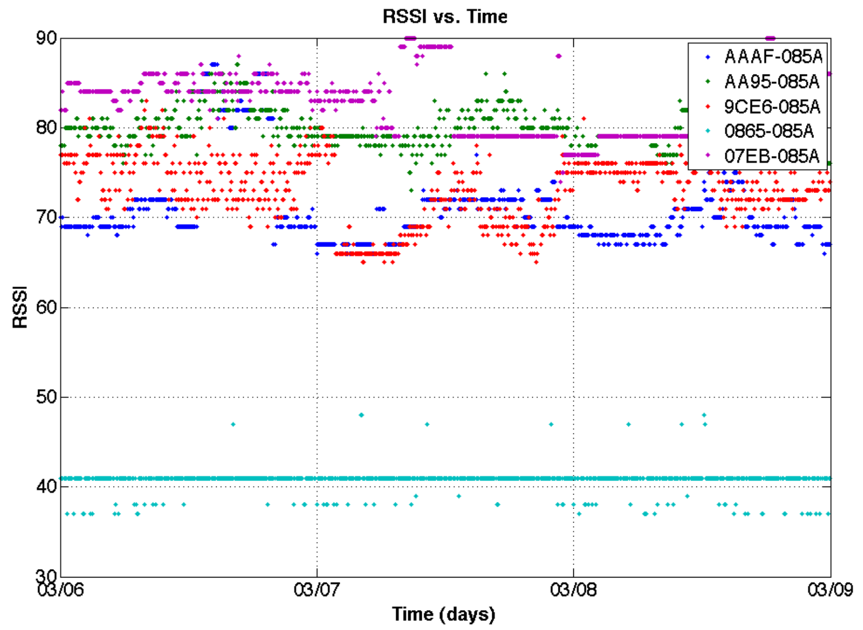


Figure 2.8: Top: links and corresponding RSSIs measured by all nodes connected to node 085A. Botttom: links and corresponding RSSIs emitted by node 085A.

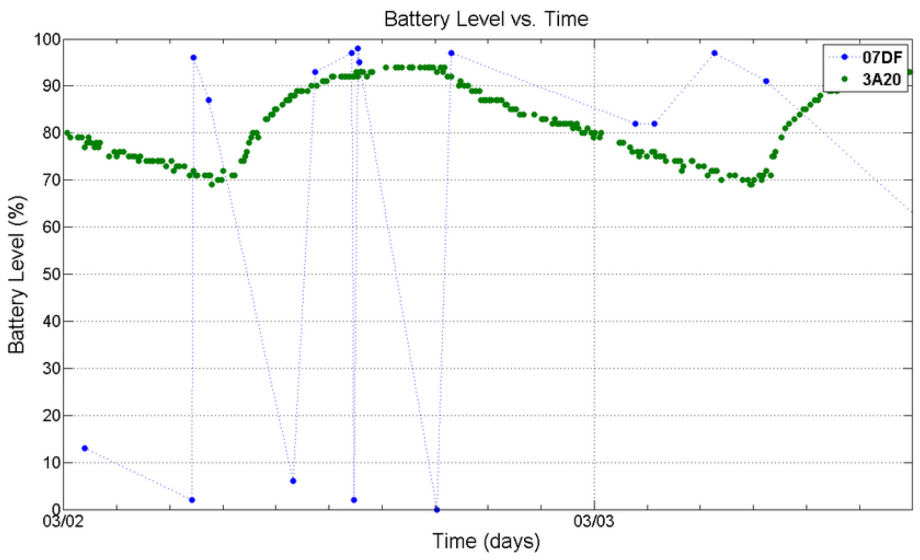


Figure 2.9: ADC failure effect on node ”07DF” during a day of experiment, compared to a working node ”3A20” with no ADC fault.

associated with all paths going through this mote.

2.3.4 Experiment planning and testing issues

Planning issues

importance of planning: Previous wireless sensor deployments experience confirm that planning is one of the most important steps [44]. Careful planning is required to determine how many nodes should be placed in a given area, and how they should be placed. Node placement is difficult, since fixed network topologies are usually limited to certain shapes such as a rectangular grid [45], [46], and cannot easily be transposed in urban environments for which relatively few mounting spots that are clear of obstacles are available.

While sink placement is usually a concern for overall network efficiency, classical optimal placement approaches such as [45], [46], or [47] need to be revised since only a few locations in cities are convenient for sink deployment.

A sink should be installed in a public place for easier accessibility and requires power and internet access. Also, our solar powered WSN deployment requires one to find spots for which both energy availability and connectivity be maximized, which makes planning more difficult particularly in the context of solar powered nodes, though these requirements are sometimes conflicting. Finally, one should avoid wireless routers as much as possible, or select operating frequencies that do not interfere with the WiFi spectrum (for instance channel F on `XBee`). Before each of our deployments, we identified all possible deployment spots, and planned a tentative deployment map. The actual deployment sites were set during the day of deployment in function of different factors, such as actual solar energy availability or unseen obstacles affecting connectivity or power availability. Scheduling deployments: Generally, the time for a node deployment tends to be initially underestimated. We assumed that we will be able to deploy 10 nodes per hour, using a flatbed truck equipped with an aerial lift. However, we only been able to reach about four nodes deployed per hour in the best possible conditions (low traffic, highly trained workers), and on average, only one mote per hour over the complete deployment, including accessing some motes twice to enhance connectivity or change solar panel orientation. Installation involved using worm gear steel clamps to fit the mote frame to the street light, using electric screwdrivers. Falling hazard was addressed through the use of multiple clamps for attachment, and by the fact that the street lights have a lower diameter at higher height.

Testing issues:

Protocols take a long time to implement and test, particularly when they depend upon a large number of parameters sent by the sink [4]. The difficulty of testing comes from many factors. First, the need to simulate an outdoor environment in the lab. Previously when we started our testing in 2011, creating sufficient light to efficiently power solar panels is impossible in the lab, since the sun sends about 1 kW per square meter, while a typical fluorescent lamp radiates 30 W over a few square meters. An additional difficulty is the simulation of multi-hops in the lab. We could not create more than two hops between all nodes and the sink even by setting the `XBee` transceivers to the lowest power setting (1 mW), and using the complete building space. Thus, we had to rely on the outdoors experiments we been able to conduct by that time. Since solar energy could not be simulated, we powered all nodes with a set of USB hubs, though this could lead to node resets whenever the USB connection is faulty.

System reliability: A very high system reliability is required in order to obtain valuable results. In the deployment of May-June 2012, for example, the network was reaching 80 operational nodes, then started to fail in an order of days before we could generate sufficient data to test the algorithm on WSN simulators (indeed, the algorithm as implemented would need a few days of data to converge). But the most important lesson of these deployments in terms of reliability is that the notion of successful deployment is a function of the duration of this deployment, and that problems that cause the network to fail can appear after days, weeks, or months. Thus a deployment cannot be called a success after few hours or days.

Parameters testing: Multiple parameter changes maybe required during testing over the period of the deployment and for a complex code such as the routing code developed by our group chances for crashes are high for the many parameter changes. The sink discovery message represented in Figure 2.10, contains some parameters, which cannot all be tested in the lab, at least not for extensive periods of time. In our case, because of parameters change overflows, we have faced crashes during outdoors tests. Therefore, it is very important to extensively test each parameter to be used for the experiment, as crashes can also be coded by undocumented limitations in the API. However, there is a tradeoff, as an error sometimes happen after a long time, while time limitation in a research setting do not allow long term tests for all parameters.

Code debugging and hardware faults: A re-occurring problem of WSNs deploy-

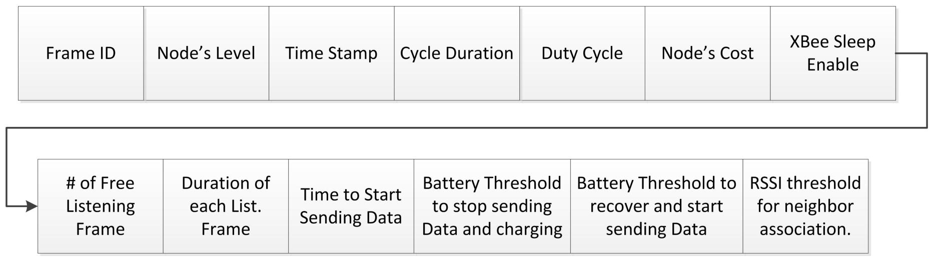

Frame ID
Node's Level
Time Stamp
Cycle Duration
Duty Cycle
Node's Cost
XBee Sleep Enable
# of Free Listening Frame
Duration of each List. Frame
Time to Start Sending Data
Battery Threshold to stop sending Data and charging
Battery Threshold to recover and start sending Data
RSSI threshold for neighbor association.

Figure 2.10: Packet structure of the Sink to initialize the network operations.

ments is the debugging and troubleshooting of the code, due to the very restricted memory and computational constraints. Dedicated debuggers might be convenient to analyze and check the execution of a code, but they will not be practical when considering large, solar powered WSN, hence we will be needing to attach a self powered debugger to all nodes. Moreover, the (`Libelium Waspmote`) platform does not contain an ISP interface, and thus extensive modifications of the boards would be required to attach a debugger, and the IDE supplied with the devices does not support code debugging such as accessing registers, or setting break points. Although debugging is always possible, it requires a large team and a very large time budget to solve all the issues, including the API bugs, particularly since some bugs are sometimes very hard to reproduce. In our case, we struggled for a few months before we understood the root cause of the hardware failures (bugs associated with the SD card libraries), since the bugs appeared rarely and randomly, over a very large network of tens of nodes that cannot be monitored completely. Also, the unavailability of database or visualization tools made the debugging problem even more cumbersome.

First, we tried to log all node activities in a file, in the SD card, but it caused random resets, due to the SD card overall unreliability. It also decreased the hardware performance in a significant way because of the high number of parameters to log. We thus relied on the serial (USB) monitor, printing all activities. However, this significantly lowers free memory, and sometimes causes more problems than it solves.

Another issue is the detection of hardware failures, which is complicated by the fact that failures usually have multiple causes, and may not fall into the mental representation of what could fail. We were for instance very surprised by the number of loose connectors detected after the May-June 2012 deployment, given that all nodes had be subjected to similar pre-deployment inspections before being loaded in the flatbed truck. The vibrations caused by the transport of nodes to the deployment sites probably caused all these connector issues, though such issues are not obvious to an electrical engineer. The failure of some `XBee` transceivers, as described in section 2.3.2, is another example of hardware fault that does not fall into the mental representation of how components can fail. We would have imagined that either the transceivers would work perfectly, or they would fail to transmit/receive data altogether. But the type of failure described in section 2.3.2 was completely outside of this mental representation: the transceivers work, but work badly and only receive a small number of messages even though the RSSI is very strong. A research group does not usually have the time to do unit tests on all components to check what can fail, and how it will fail. These issues have a considerable impact in practice, dramatically slowing down applied research and implementation.

2.3.5 Accessibility issues

<u>The importance of nodes accessibility</u>:

To maintain a WSN, the ability to access nodes is very important. Nodes locations may need minor adjustment for better connectivity or higher solar energy availability, even in the absence of bugs. However, these accessibility constraints are incompatible with urban deployments because, we want to maximize node wireless connectivity ranges to maximize the network coverage, and we thus need to mount the nodes high above the ground, at an elevation of more than five meters. Furthermore, local police and urban services do not want to install anything that can be easily tampered with by people for liability and safety reasons, which prevents easy node access. Unlike deployments in unpopulated areas [43], the monetary cost of accessing a node in an urban environment is very high, and faults are expensive to fix.

Over the air programming:
Over the air programming (OTA) allows the firmware to be remotely upgraded in a node. This feature is absolutely mandatory for long term deployments in urban environments, in which nodes can be hard to access. While the platform used for this study (`Libelium Waspmote`) supports OTA, the version implemented in 2012 was highly unreliable, and caused numerous crashes in lab experiments (this was probably caused by the lack of reliability of the SD card, since OTA uses the SD card to store the firmware during code update). The lack of an OTA means that the code must be absolutely perfect and thoroughly tested, which conflicts with a research agenda.

Changeable network parameters:
We had to leverage the changeability of our packet structure of the discovery message sent by the sink (see Figure 2.10) to modify the network parameters since we could not rely on OTA to update the code. However with this method we cannot yet test a completely different code and the parameters used for the network should be extensively tested in the lab as overflows can happen and cause irrecoverable crashes.

2.3.6 Deployment challenges

Routing protocol startup:
The complete discovery procedure with route propagation, energy propagation and RSSI propagation taking slightly less than one hour which required all nodes to be installed before they could be discovered by a signal from the sink. The nodes would reset every 24 hours, waiting for a new discovery signal from the sink. However, this caused problems, as the energy consumption of a node while listening is very important, with less than two days of theoretical endurance in this mode. This

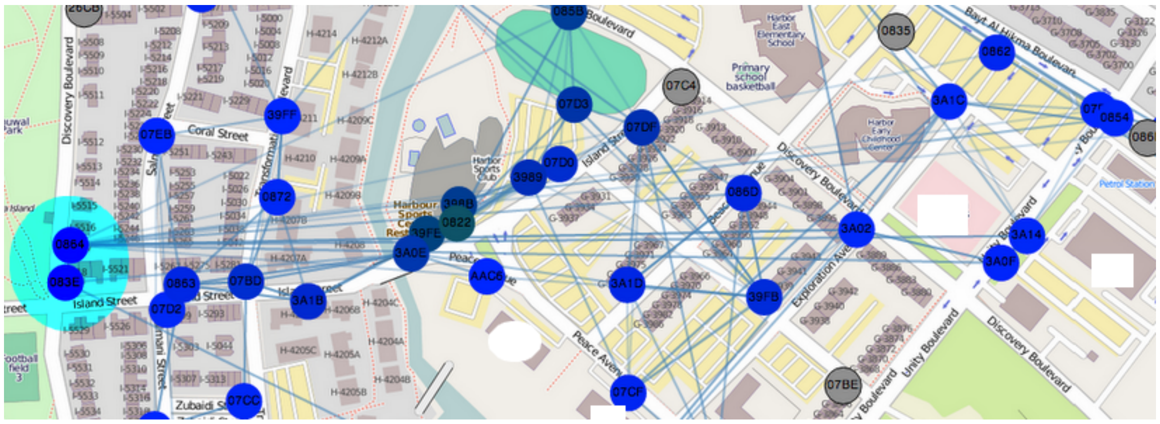


Figure 2.11: Two nodes placed incorrectly, i.e. 0864 and 083E (in light blue)

approach also prevented us from immediately validating if the node was actually turned on, and in range of the rest of the network, leading to a relatively large number of nodes failing to join the network (only 52/80 were part of the network). Subsequent tests rely on the SD card-free version of the algorithm, which does not require a long discovery and allows nodes to join the network very quickly, enabling an immediate confirmation that the mote is indeed working, greatly facilitating deployments.

Node mis-labeling:

Labeling nodes becomes a very essential step to properly establish the structure of the map of the network. In the early PhD implementations, we used integer numbers as labels, but this required us to build a one to one map between nodes and transceiver MAC addresses. We thus decided to label nodes using the four last characters of their MAC addresses, which are unique in the present case. Though this helps avoiding mislabeling or node misplacement, node misplacement on the map sometimes occurs, as illustrated in Figure 2.11, which shows two nodes "0864" and "083E" that we immediately suspected to be misplaced (based on their very strong, long radio links). This misplacement was later confirmed after node removal. Again, node inaccessibility prevents one from easily confirming this problem without relying on an expensive aerial lift.

During the early tests, it was common to just place the nodes around campus at a low level, to test the multi-hop routing protocol. Since the tests sometimes lasted a few hours, we printed warning labels on them (see Figure 2.1) to avoid people tampering with the devices. This did not completely prevent curious people from accessing the devices, as in one instance a `XBee` transceiver was found unplugged near a node. Installing motes in inaccessible locations to public is highly recommended.

2.3.7 Data analysis issues

Database setup issues:

Databases are easy requirement to set up in WSN applications, and both the quantity of data and the data rate are not causing any source of issues. However, maintenance is an issue when multiple deployments are considered, since not all nodes are used for all deployments, in variable positions. We used a properties table for the nodes in addition to tables containing data. We previously had RSSI, energy and link traffic as main parameters. Each experiment required the definition of new data tables, though the properties table might not change if nodes positions are similar. While efficient in terms of storage, this approach requires us to keep track of which table is associated with which properties table. To solve this, we now have single data tables that contain geospatial information, i.e. each data point generated by one node is associated with its lat. and long. coordinates for example. While this is relatively inefficient, it is not too much of a practical problem since the database is absolutely not constrained in terms of storage. It is also much easier to playback previous experimental results for comparison.

Database interfacing issues:

`MATLAB` was used to handle the packets received by the sink and fill the respective tables. Besides the classical internet outages, the biggest issues were data handling. The first issue is the possibility for an `XBee` to receive corrupted data (mostly serial read issues), which leads to problems when parsing the data. Try/catches have to be used everywhere in the code. The second issue is the relatively high data rate at the sink level, which often causes the `XBee` to be temporarily unable to respond before recovering (during this time, all incoming data is lost). We address these issues by periodically resetting the transceiver.

Visualization:

To visualize data, a visualizer was built using the Pyramid - a python web framework on the server side and d3.js on the client side. Data is read from the database and processed on the server side prior to being delivered to the clients as a JSON document.

The easiest way, however, to visualize data nowadays is to leverage map generator websites such as `cartoDB`, or `ArcGIS` on which we are currently porting our visualizer.

2.4 Experimental investigation of environmental perturbations on the WSN operation

Sensing large scale environments can be achieved through wireless sensor networks (WSNs), which consist in a wireless mesh of sensing nodes. A large number of applications of WSNs are in outdoor environments, in which the environmental parameters have considerable variability, unlike indoor environments that are designed for human habitation. Being physical systems, wireless sensor networks are impacted by these environmental parameters, which can negatively affect their performance. Quantifying this performance degradation is very important in practice, as network control schemes and energy management schemes have to take these effects into account.

2.4.1 Experimental setup

The experiment involved 34 nodes and started by late February 2013, lasting for two months. To study the relation of the temperature and irradiance effect on Received Signal Strength indicator RSSI and Packet Delivery Ratio PDR we decided to do

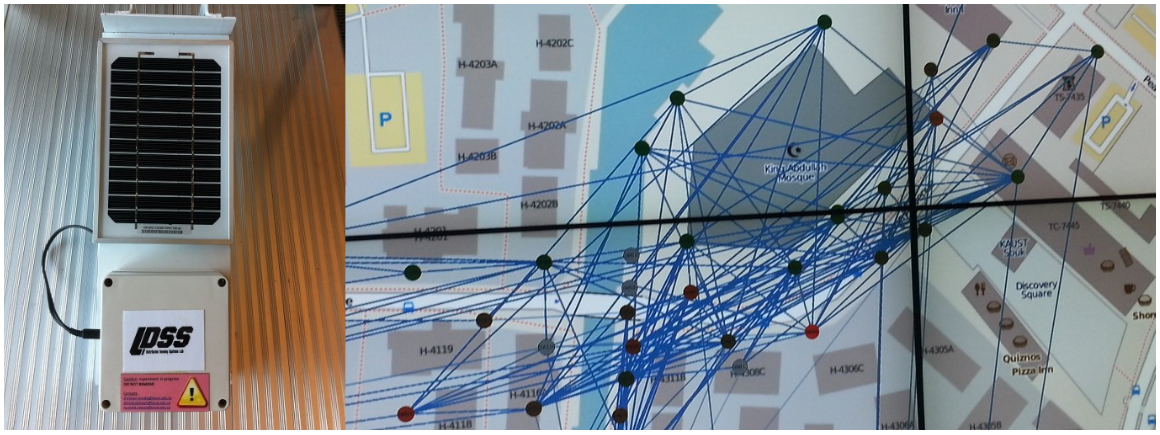


Figure 2.12: One unit node (left) used in this analysis and sensor network visualization (right) for the 34 nodes

this study over eight consecutive days (between two gateway failures) starting March 1st 2013. In this dataset temperature and humidity both have short term periodic (daily) and long-term (weekly) variations. We use as the ground truth data the data generated by the local solar power testing station, which generates solar panel temperature, irradiance and humidity measurements every 900s. The ground truth meteorological data is illustrated in Figure 2.13.

2.4.2 Link performance analysis

Impact of humidity:

There is significant debate in the literature on the actual effects of humidity and temperature on link quality in sensor networks. Thelen et al. [48] claims that a higher relative humidity improves the received signal stength, while Anastasi et al. [49] conclude that fog and rain cause a decrease in packet reception ratios. Interestingly, these findings contradict the fact that radio signals on frequencies below 11 GHz should be unaffected by fog and rain, as they are transparent to water.

To distinguish the individual effects of the environmental parameters on link quality, we did two multivariate regressions on both link RSSIs and packet reception ratios between nodes, and show the results in Figure 2.14

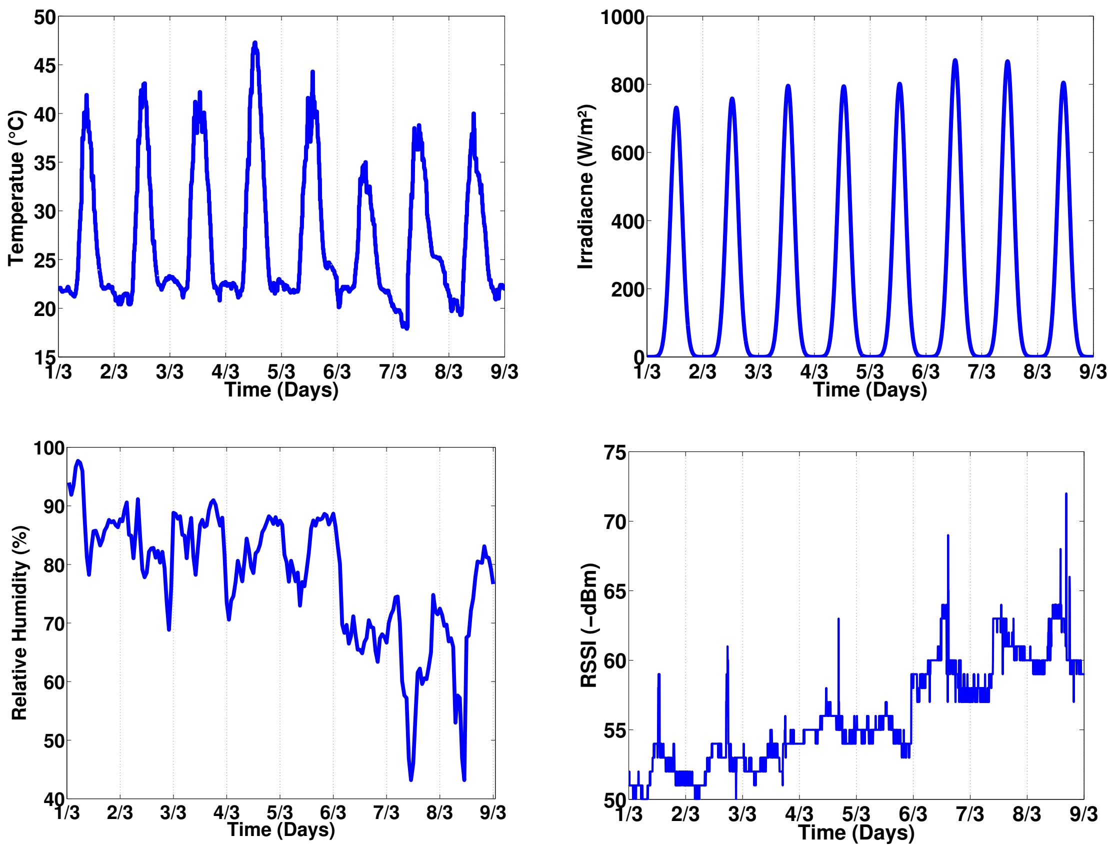


Figure 2.13: Temperature (top left), irradiance (top right) and relative humidity (bottom left) measured by the weather station. RSSI average timeseries for a representative link (bottom right)

Impact of temperature:

While we can measure the solar panel temperature accurately, the mote temperature can significantly differ from the solar temperature due to air cooling and to internal energy dissipation. In first approximation (neglecting blackbody radiation), the temperature increase of the enclosure of area S, thickness d and thermal conductivity κ subjected to a power forcing P is $\delta T = \frac{P \cdot d}{\kappa \cdot S}$. In the present case, the motes (which dissipate 0.2W) are 0.1C hotter than the panel temperature (which is equal to the air temperature) during the night, and can be up to 7 degrees cooler than the panel temperature with an (irradiance) of $800W/m^2$ (when they are in shadowed areas).

The effect of the temperature on link quality is shown in Figure 2.14 below.

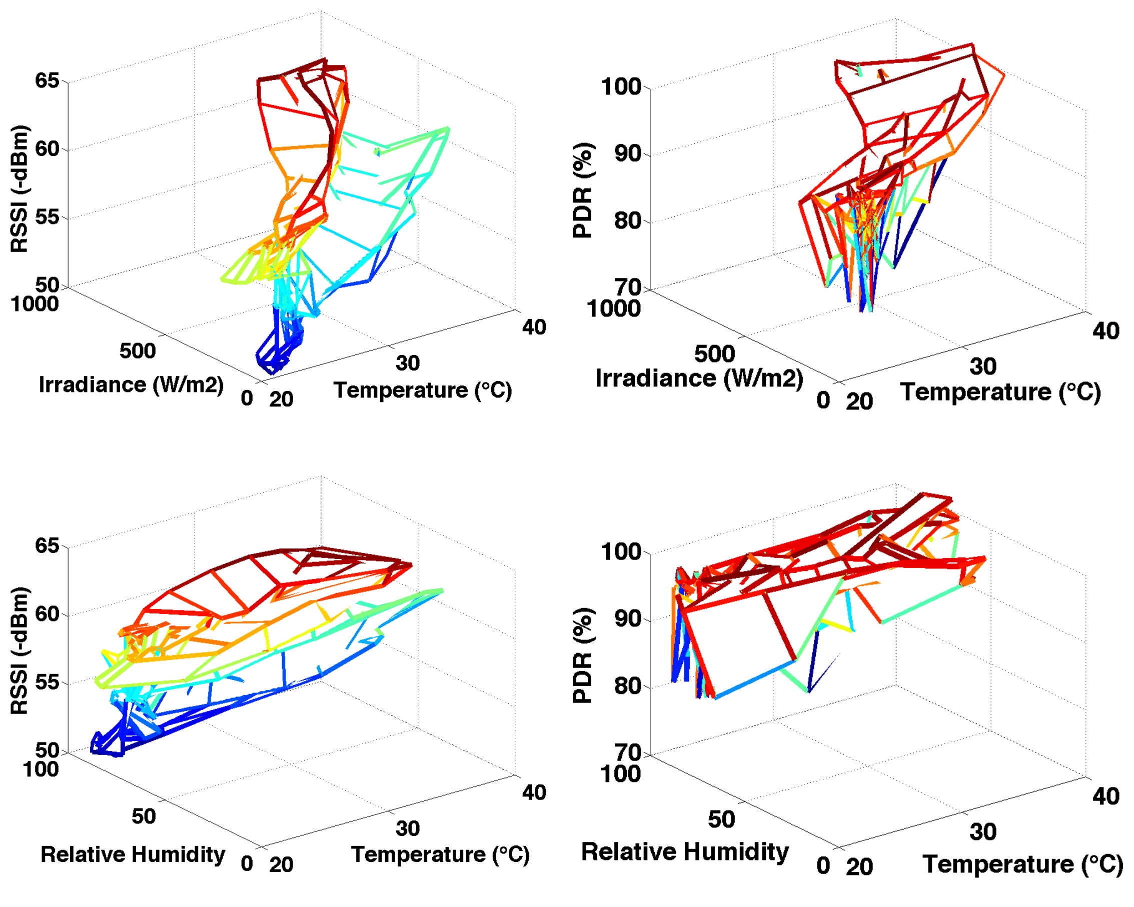


Figure 2.14: Environmental effects on link quality. Top left: RSSI versus panel temperature and irradiance. Top right: packet delivery ratio versus panel temperature and irradiance. Bottom left: RSSI versus panel temperature and humidity. Bottom right: packet delivery ratio versus panel temperature and humidity.

As can be seen from the above Figure, temperature has the largest impact on link quality. In fact, a multivariate regression on temperature, irradiance and humidity gives the following results:

$RSSI = \alpha \cdot T + \beta \cdot I + \gamma \cdot H$, where $\alpha = 95$, $\beta = -35$ and $\gamma = 11$. The uncertainties (from the covariance matrix) are $\Delta\alpha = 9$, $\Delta\beta = 4$ and $\Delta\gamma = 5$. Thus the RSSI is very strongly correlated with panel temperature, negatively correlated with irradiance and statistically uncorrelated with humidity. The negative correlation with irradiance is expected as air temperature is positively correlated with panel temperature and negatively correlated with irradiance.

Temperature affects link quality since it increases the level of thermal noise (which in turn reduces the probability of successful decoding).

2.4.3 Power analysis

Power consumption:

Since our motes are not currently equipped with battery charging current sensors, we have to estimate their consumption by analyzing the battery voltage drop over time during the night (when no solar power forcing is present). Since the voltage-charge relationship in a battery is dependent on the battery condition and discharge rate, we estimate the battery condition using the procedure outlined in [50]. Results are shown in Figure 2.15 below.

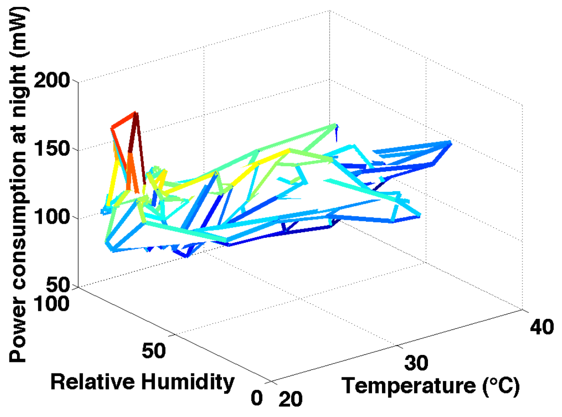


Figure 2.15: Impact of environmental effects on estimated night power consumption. Power drawn from the battery versus air temperature and humidity.

A similar multivariate regression gives the following results:

$Power = \alpha \cdot T + \beta \cdot H$, where $\alpha = 0.5$ and $\beta = 3$. This shows the limits of multivariate linear regression, as our data is far from being normally distributed in the present case. Thus the dependency with the humidity is mainly driven by a few outliers for extreme humidity values. Power dissipation is approximately constant over the $40 - 90\%$ humidity range.

Power dissipation increases with temperature since the resistivity of semiconductors decrease with temperature, causing higher current drain (at the same voltage). The outliers appearing in extreme humidity conditions could be caused by condensation of water on the electronics in some motes due to enclosure damage.

Chapter 3

Energy Estimation for solar-powered WSN in desert environments

A large part of the cost of WSN is the deployment installation, specifically if it requires to be connected to a certain power grid. Hence, the aim of our research to provide a cost efficient WSN, we wanted to investigate self powering options. For application, such as smart parking or traffic flow monitoring where low power is required, batteries can be used, but the wireless range would become substantially reduced, which in turn will require more nodes to be installed as relays, increasing the cost. Therefore, harvesting energy from the environment becomes a very vital asset, and enables extended communication, sensing and computational capabilities. In urban environments (which is our area of concern), where sensors would be placed in lower attitudes, solar power appears to be the most reliable and efficient source of energy for our applications compared to other possible energy sources. However, using solar power to operate WSNs comes with great challenges especially when the energy generated by the solar power is not significantly greater than the current draw, which is the case for high power monitoring applications.

In this chapter, I carried on the post deployment analysis done in Chapter 2, for one of the deployments starting November 2012 (particularly deployment 3, in section 3.2.2), presenting a medium-scale experiment with 34 nodes. Our nodes are powered both by a solar panel in-conjunction with a rechargeable Lithium ion battery. This set of experiments showed the importance of investigating the energy storage and power availability parameters that are significantly varying. We realized two key areas of concern to solve the energy estimation and prediction problems:

- Battery capacity and condition estimation. We concluded after our experiments that the conditions associated with the batteries is highly correlated on discharging behaviors and environmental effects.

- Power availability estimation. We concluded from our experiments that the power availability in our WSN nodes vary in a great way, despite that the deployment was carried out in the same geographical area. For sensing application, this fact becomes of great importance hence sensing duty cycle of sensing can be adjusted based on the available and predicted energy.

For energy-aware WSNs, these identified energy parameters shall serve as inputs [51, 52, 53, 54]. Thus, we investigate the estimation of these parameters for efficient operations. The focus of this part of the thesis is to design a suite of energy estimation that can all be implemented in typical WSN microcontrollers.

3.1 Energy management state of the art

power management is an active research field in WSNs, as in energy monitoring in [55].The deficiency of dedicated monitoring systems in existing platforms is one of the main challenges appearing with energy estimation, which may require the development of dedicated hardware and software [56, 57]. In [58], the Lithium Ion capacity depletion and charging is modeled to forecast power availability, though some important parameters, typically battery capacity are not actively estimated. Many researchers [59] consider some electrochemical phenomena, such as capacity rate characteristics, charge recovery and thermal effects, which can play a role in governing the selection of sensing parameters. In this work, our objective is to analyze the energy patterns in nodes subjected to a external source, in a harsh environment (humidity, dust, temperature), and to propose computationally efficient methods for estimating energy and power parameters dynamically. The resulting algorithms are to be implemented in the nodes themselves, which have limited computational capabilities.

3.2 Experimental Setup

3.2.1 System

Our proposed system consists of an heterogeneous wireless sensor network for monitoring traffic flow in cities using both fixed and mobile sensor data. The speciality of this sensor network lies in its monitoring operation: the flood and traffic state estimates are directly computed by the wireless sensor network, and are forwarded to an centralized database. This distributed computing based operation allows the system to be cost efficient and easier to deploy, since no costly redundant estimation servers and input databases would be required. It would also enhance user privacy, since user data would remain local (only traffic monitoring parameters and flood forecasts would be global).

The platform chosen for this analysis is described previously in Chapter 2, in section 2.2.1.

The `Waspmote` can be interfaced with two types of transceivers from `XBee`: one transceiver implements the `Zigbee` protocol, while the other implements IEEE 802.15.4 standard. While `ZigBee` handles node synchronization and is relatively energy efficient, it cannot be used in the present application, as it would require coordinators that have to be always on (this cannot be guaranteed in practice for a solar powered wireless sensor network). In addition, a coordinator is a single point of failure, and could potentially render a large portion of the network inoperative. We thus chose IEEE 802.15.4 `XBee` transceivers, which require the development of a routing protocol supporting multi-hop communication.

For this part of our work, we used the standard libraries provided by `Libelium`, though we commented all unused functions from the API to increase the available memory. We also added a hardware reset circuit to enable regular node reset (explained in section 2.3.1), improving overall reliability.

3.2.2 Energy estimation deployment

We installed 34 nodes on both sides of a street, in an urban environment in the campus of King Abdullah University, in area of Thuwal starting November 2012. The maximal distance between any two nodes in the network is approximately 400 meters, as illustrated in Figure 3.1 for this particular deployment. The network has a radius of 5 to 6 hops, depending on the presence of parked vehicles around the main road, as well as other environmental conditions. Each node transmits its energy and local connectivity map every 300s (cycle time). The resulting data can be displayed on a web-based visualization designed using `The Pyramid` (a `Python` web framework), and is also stored in a database (`PgAdmin`).

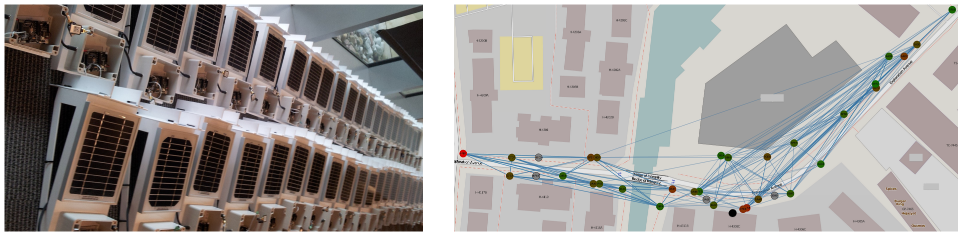


Figure 3.1: Left: WSN nodes used in this study. Right: Web-based visualizer dedicated for this energy estimation WSN deployment.

3.2.3 Experiments results

A subset of the battery voltage timeseries is illustrated in Figure 3.2 below. The data collected at the sink were in voltage values between 2.7V and 4.2V, scaled 0-100% in the figure, each mote being represented by a different color. ADC failures (addressed in section 2.3.3) can also be inferred from this figure. One month of experimental data has been used in our analysis.

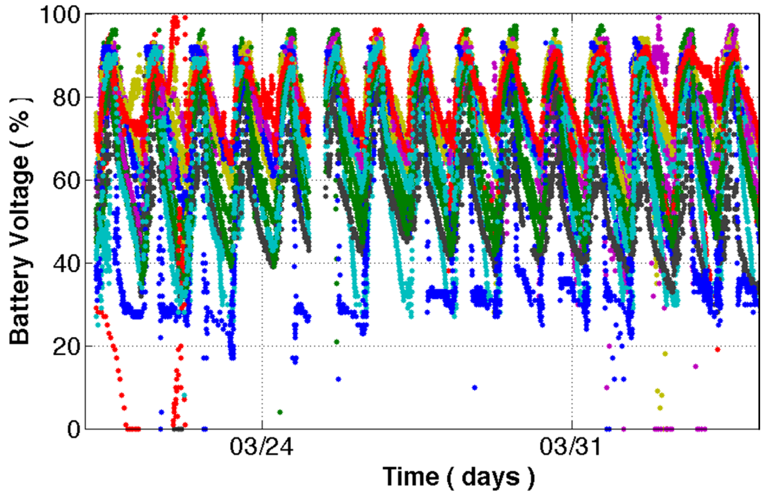


Figure 3.2: Energy timeseries from 34 nodes between March 21st and April 4th, 2013. The daily energy charge-discharge cycles are clearly visible.

3.3 Energy Model

3.3.1 Energy Generation and Storage

Each node is equipped with a rechargeable battery, connected to a solar panel through a charging chip. The complete energy management system is shown in Figure 3.3 below.

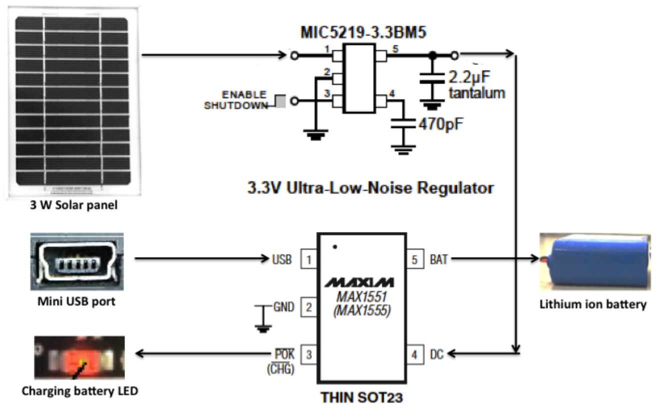


Figure 3.3: Energy management system circuitry in a solar-powered node.

The solar panel used for this study has a 3W peak rating, which implies that it can generate at most 3W of electrical power.

The charging chip efficiency ζ is the ratio of its output to input electrical power, and is around 90% in the present case. The battery used for this study is a 2300 mAh 1-cell Lithium Ion battery.

3.3.2 Energy Conservation Equation

Each node in the WSN broadcasts one update containing its energy, RSSI and link statistics with other nodes every 300 seconds (5 minutes).

All nodes are programmed with an identical software, and form a multi-hop mesh network. The average power consumption of the `XBee` transceivers in listening mode is 165 mW, while their power consumption in sending mode is 660 mW. These figures are a function of the ambient temperature, a higher temperature being associated with a higher power consumption (see also section 2.4.3).

Since our objective is to study the energy evolution in a wireless sensor network, we configured the sensor network in such a way that the power consumption is approximately the same for all nodes. The 300s cycles consist in a listening phase of 150 seconds, during which each node sends between 1 and 34 (worst case number of messages to relay through multi-hop) messages. Thus, the total energy consumed by a node during a cycle is between 24.8 J and 25.4 J (we assume that the data rate is 32 kB/s, and that each packet is 100 bytes including overhead). Since power consumptions differ by 2.5% at most, we can assume that the overall power consumption is independent from the node, particularly since the temperature differences between nodes (caused by cast shadows) cause changes in the power consumption that are higher in magnitude. The average power draw of the transceiver is thus 85 mW, which, together with a power draw of 30 mW from the microcontroller yields a total average power consumption of 115 mW.

Let $E(k)$ be the energy stored in the node battery at (discrete) time kT. The conservation of energy [60] can be written as:

$$E(k+1) = E(k) - P_{draw}T + P_{solar}T \tag{3.1}$$

Where P_{draw} and P_{solar} represent the average power draw and power generated by the solar panel respectively. One of the constraints of this system is the fact that the power generated by the solar panel is not directly related to the power density of the sunlight (irradiance) received by the solar panel, because of factors related to the chain of energy conversion itself (Figure 3.3). First, the solar panel and charging chip efficiencies are a function of their operating voltage, which cannot be adjusted (in particular the solar panel is not necessarily operating at its voltage of peak efficiency). Second, the power that can be actually transferred to the battery is a function of the battery condition (capacity, internal resistance) as well as its current charge (see Figure 3.2 and Figure 3.5). In our solar power estimation problem (see Figure 3.5), our objective is to estimate the maximal available solar power, assuming that this power could be transferred to the battery.

3.3.3 The Need for Energy Estimation and Forecast

Despite the cyclic nature of the sun irradiance, the energy availability in a solar-powered wireless sensor network will significantly vary between nodes, for a variety of reasons:

- Large scale weather effects (uneven distribution of solar energy).
- Short scale urban shadow effects (cast shadows), these effects can appear during specific seasons (winter) depending on the latitude and the sensor network deployment locations.
- Debris, dust, humidity and salt accumulation (which are a function of the solar panel orientation).

- Battery condition and remaining capacity (which is a function of the battery history, in particular the number and the severity of deep discharges). Since a battery that has a lower capacity is more likely to be discharged deeply during its future operation, the differences between batteries tend to get worse over time.

To enable energy aware sensing, computing and communication schemes [61], it is critical to be able to accurately forecast future (medium-term) power availability, and to estimate the current battery condition. With such parameters, the operation of the system can be scheduled in real time as a function of expected energy availability and current energy storage performance. Because of the short term (daily), medium term (seasonal) and long term (rain, dust, debris, battery and solar panel degradation) variations in energy supply, this estimation and forecast process has to be conducted relatively frequently.

Estimating the power availability in real time has additional benefits, for instance to detect faults in the battery, solar panel or charging chip. This early detection allows the optimization of the remaining energy of the node to allow enough time for its repair.

3.4 Estimation of Battery Condition and Capacity

3.4.1 Background

As all batteries, the Lithium-Ion batteries used as an energy buffer in all nodes tend to age during their operation, which translates into a loss of capacity (or as an inability to fully charge the battery) and an increase in internal resistance.

Single-cell Lithium-Ion batteries (such as the battery used in this part of the work) have a maximal voltage of 4.2 V, and a minimal voltage of 2.7 V, below which a battery protection circuit prevents further discharge. In general, the loss of performance of a battery [62] depends on the following factors:

- Environmental conditions (in particular the temperature)
- Extended operations at full charge
- Depth, duration and number of discharge/charge cycles

Though no battery datasheet was available to us, we used standard industry figures to evaluate the battery performance. At the beginning of this test, all batteries had been used for less than 100 charge/discharge cycles. Industry figures indicate that batteries can handle between 400-1200 cycles before losing 30% of their original capacity, though these values are highly dependent on environmental conditions. Therefore, our batteries can be considered as relatively new. The total duration of the experiment (3 months) added less than 100 cycles to these batteries history. The ambient temperatures measured during the experiment were comprised between 15 °C and 35 °C, which fall within the recommended operating temperature range for these batteries.

As stated in Section 3.3, the discharge rate during listening operations is on the order of 200 mW, while the maximal current that can be delivered by the charging chip is 280 mA, which is well within the recommended values of 1C (2300mA) for charge and 2C (4600 mA) for discharge of standard Lithium-Ion batteries.

In this work, we excluded motes for which the voltage timeseries were inconsistent with the physics of the system (which denotes an ADC fault), for example when the voltage variations between two consecutive points exceed the maximal power that can be delivered by the charging chip or the maximal power that can be dissipated by the battery. ADC failures can be directly inferred from Figure 3.2.

3.4.2 Estimation of Battery Discharge Patterns

We focused our investigation on only the data generated during the night (*i.e.* after the sunset and before the sunrise), in which the solar panel generates negligible power (the illuminance caused by urban street lights is orders of magnitude much smaller than the full daylight illuminance).

The battery charge decreases linearly during the night since the current drawn by the microcontroller and the peripherals is constant (approximately 35 mA). The voltage data collected during the night thus provides us information on the relationship between battery charge Q and battery voltage V. Since the loss of energy during one night does not span the complete operational range of the battery voltage, we integrate the data of multiple days to estimate the discharge curves Q(V), where Q is determined up to a constant (this does not impact our analysis however, as we are interested in the derivative of Q and not in Q itself).

We first define $t_i(V)$ as the inverse of the nighttime battery discharge timeseries $V_i(t)$ associated with nodes $i \in [1, n]$. Since the current drawn from the battery is constant[1], we have that $Q_j(t) = Q_{0,j} - i_0 \cdot t$, or equivalently $t = \frac{Q_{0,j} - Q_j(t)}{i_0}$. Given that the $Q_{0,j}$ (initial charge) are unknown for all j, we fit these functions by solving the following least squares problem:

$$\min_{Q_{0,1},\dots,Q_{0,n}} \sum_{i,j,i \neq j} \int_0^{V_{\max}} \left(t_i(V) - t_j(V)\right)^2 dV \tag{3.2}$$

Problem (3.2) is an unconstrained quadratic programming(QP) which can be solved using standard linear algebra. We show the results of the above discharge curve fitting scheme for one node over two different time windows of 10 nights in Figure 3.4. For these graphs, we assumed that 2.7V (which is never attained in practice) corresponded to a zero charge. Note that most circuits will not discharge a lithium

[1]This hypothesis assumes that the current drawn by the microcontroller and its peripherals is independent of their temperature.

ion with a voltage less than 2.7V to avoid irremediably damaging the battery.

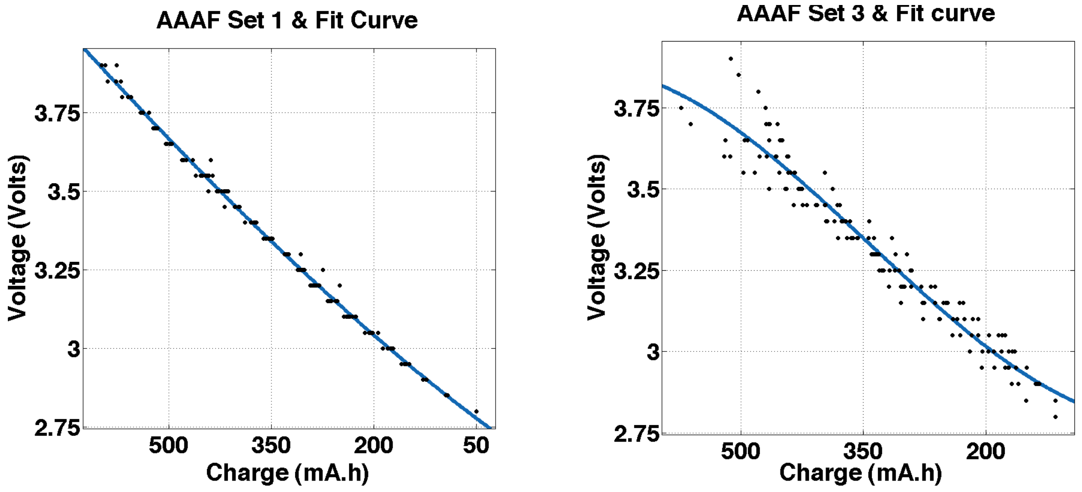


Figure 3.4: Discharge curves of batteries AAAF evaluated on a 10 days period at the beginning of the experiment (Left), and on a 10 days period 20 days later (Right). The right subfigure shows a slight degradation in battery capacity, as well as a less reproducible (higher standard deviation) discharge pattern.

3.4.3 Estimation of Battery Capacity

The battery capacity is estimated from the charge of the battery when its maximal voltage is reached (which is a function of the battery, see Figure 3.2). The pseudo-code used for the above battery capacity estimation is presented in Algorithm 1 below. This codes relies on a polynomial fit (we used a polynomial of order 3 in practice) to estimate the charge at 2.7V.

Algorithm 1 Pseudo-code implementation of the battery discharge curves and battery capacity estimation Algorithm.

```
Require: $V_i(t), t \in [t_{\text{sunset,i}}, t_{\text{sunrise,i}}], i \in [1,n]$   {Voltage timeseries for $i^{th}$ night, $i \in [1,n]$}
  for $i = 0$ to $i = n$ do
    Low pass filter($V_i$)$\rightarrow V_i$
    $t_i(\cdot) := V_i^{-1}$
  end for
  $\min_{Q_{0,1},\dots,Q_{0,n}} \sum_{i,j,i\neq j} \int_0^{V_{\max}} (t_i(V) - t_j(V))^2 \, dV \rightarrow Q_{0,1}, \dots, Q_{0,n}$
  return $(Q_{0,1}, \dots, Q_{0,n})$ {Fitting parameters for discharge curves}
  Polynomial_fit($\cup_{i\in[1,n]} Graph(Q_{0,i} + i_0 \cdot t_i(\cdot))$) $\rightarrow P(\cdot)$ {Polynomial fit of the reconciliated discharge curves}
  return Capacity $C = P^{-1}(\max_{t,i} V_i(t)) - P^{-1}(V_{\min,battery})$ {Estimated battery capacity, $V_{\min,battery} = 2.7V$ for Li-ion batteries}
```

Table 3.1: A systematic analysis over 9 nodes yields the following estimated capacities.

Mote ID	AAA8	AAAF	9CE6	AA95	0865	07EB	085A	3A02	0897
Capacity (March 1-10)	525	575	700	665	740	595	548	397	784
Capacity (March 11-20)	530	566	712	630	712	562	520	377	780
Capacity (March 21-30)	528	545	620	603	686	553	490	345	710

As one can see from the above Table 3.1, the variations in capacity are significant with capacities ranging from 350 mAh to 800 mAh. Note also that the estimation is fairly robust, with only minor discrepancies between battery capacity estimates. All estimated capacities are less than one third the original battery capacity, which show that Li-Ion batteries degrade quickly in outdoor environments. For the coming explained research in this Thesis, we used Lithium Iron Phosphate batteries, which have a slightly lower energy density, but a much greater tolerance to temperature fluctuations and deep discharges.

3.5 Estimation of Solar Power Supply

While the battery condition and remaining capacity are a very important factor for energy management of WSNs, the solar power availability is an equally important parameter. The remaining capacity is a measure of the potential energy that can be stored in the battery, while the solar power availability can be thought of as a measure of the actual power input.

To estimate the availability of solar power, we first estimate the current generated by the solar panel during the day (similarly as in section 3.4) from the battery voltage time series obtained during the day. The actual power input is then inferred from the product of current and battery voltage. Two examples of solar power input are shown in Figure 3.5 below.

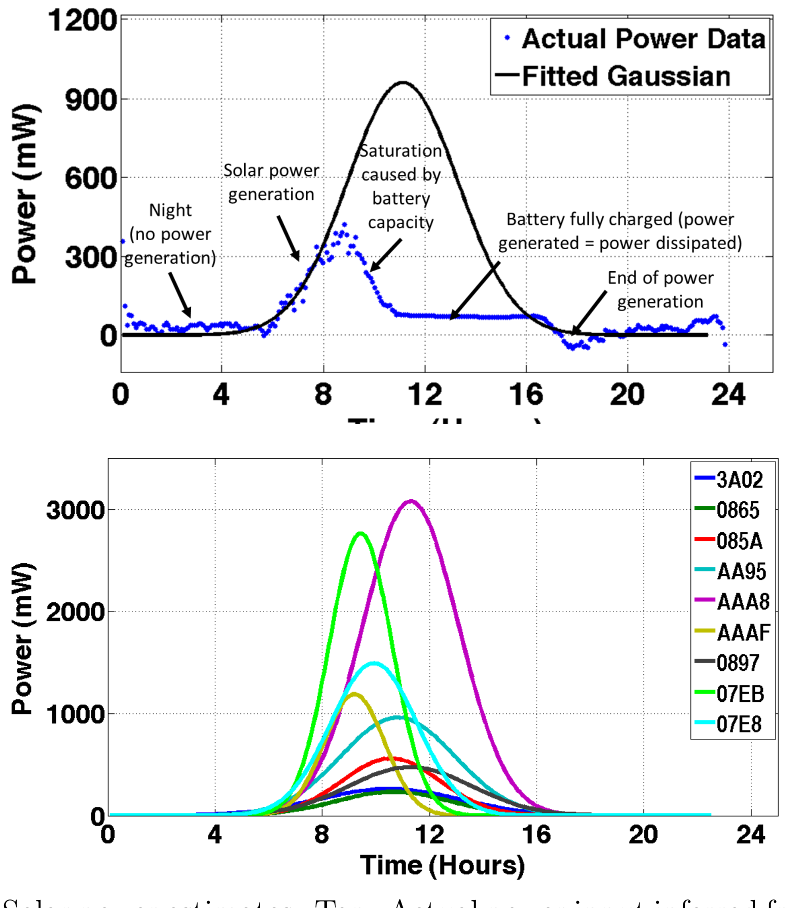


Figure 3.5: Solar power estimates. Top: Actual power input inferred from Algorithm 2 on node AA95, and Gaussian fit of the estimated solar power available. Bottom: Gaussian fits of the estimated available solar power on a subset of 9 nodes, during a typical day (March 1, 2013).

Figure 3.5 shows three main regimes of operation of a node:

- During the night, no solar power is generated (though the estimated generated power exhibits some noise). This corresponds to the times 0-6 and 17-24 in Figure 3.5, up.

- After sunrise, some solar power begins to be generated. The solar power peaks and then reduces when the battery is close to being charged (times 6-11 in Figure 3.5, up). At the beginning of this charging phase the solar power generated by the panel is close to the maximum that a solar panel can generate, provided that the battery charge is much lower than capacity, and that the generated power is not limited by the charging chip specifications.

- The power generated is then constant, equal to the power consumption of the mote when the battery is close to being charged (Figure 3.5 from 11-17). During this phase the power that could be generated by the solar panel is greater or equal to the power consumed by the node (with equality arising around 17:00 in Figure 3.5, up).

Following [63], we use a Gaussian irradiance model, which has only three parameters, and can be easily fitted by the mote (this results in a three dimensional least squares problem, though simpler methods can also be used). Gaussian fit is quite accurate for our study with low complexity which require less computation at the node level. The pseudo-code used for the fitting is shown in Algorithm 2 below.

Algorithm 2 Pseudo-code implementation of the solar power estimation Algorithm.

```
Require: $P(\cdot), V_i(\cdot)$ {Polynomial fit of the Q(V) function, Voltage timeseries for $i^{th}$ day}
  Low pass filter($V_i$)$\rightarrow V_i$
  Define $Q_i(\cdot) = P(V_i(\cdot))$
  Beginning and end of charge cycle detection (thresholding on $Q_i'$) $\rightarrow (t_{b,i}, t_{e,i})$
  Charge current saturation detection (zero of $Q_i'$) $\rightarrow t_{s,i}$
  Define $Pw_i(\cdot) = \frac{Q_i(\cdot)\cdot V_i(\cdot)+Pw_0}{\zeta_{\text{charging chip}}}$ {Solar panel input power}
  Gaussian_fit($Graph((Pw_{i|[t_{\min,i},t_{s,i}]\ cup[t_{e,i},t_{\max,i}]}))$) $\rightarrow R(\cdot)$ {Gaussian fit of the restriction of $Pw_i$ to the non-saturation domain}
  return $R(\cdot)$ {Fitting parameters for the solar power curve}
```

Figure 3.5 bottom shows Gaussian fits of the maximal solar power available to 9 different nodes during March 1, 2013. As one can see from this figure, there are significant differences in solar power availability between nodes. These differences are caused by a set of factors including orientation, dust/humidity accumulation or the presence of clouds or shadows. Note that the latter factors cannot be captured by the Gaussian fit (which assumes a direct view to the sun), and would require more sophisticated models to be used.

An important assessment of the validity of the method comes from the fact that the estimated maximal power generated by the solar panel is always below its 3W rating. Again, because of the charging chip input current limitation, not all this estimated power may be useable in practice.

3.6 Chapter Remarks

This energy estimation work presents a set of tools for estimating the energy performance of solar-powered WSN deployed in urban environment operating in desert-like conditions . Given the variability of solar energy availability and of the available battery capacity in a typical sensor network, we stress that such tools will be very useful to optimize sensing and network operations and to minimize the number of nodes that run out of energy. While low power wireless sensor network do not necessarily need such an optimization (provided that their solar panels and batteries are oversized) to run, though this suite of tools is very important for fault detection and isolation, by comparing energy and power availability with adjacent nodes.

Future work in this part of the thesis will deal with the implementation of the above methods in the nodes. All methods introduced in this work can run on typical 8 bit microcontrollers, since they mainly rely on basic thresholding and least squares fitting, which can be done efficiently with matrix operations. The required memory is on the order of hundreds of bytes, and the execution time is not expected to be an issue as computations are to be performed once daily. Also, Our analysis could be extended with regard to solar power supply for the whole data for different days to conclude with similar patterns.

Now that the energy estimation problem is addressed, the research objectives can be dedicated towards the applications and their estimation algorithms in a more efficient and clear way

Chapter 4

Flash Flood and Rain Monitoring and Detection Using Ultrasonic and Infrared Sensors (Estimations and Algorithms)

Wireless sensor networks (WSNs) are widely used for monitoring and control applications [64], [65, 66], [67, 68, 69] such as environmental surveillance [70, 71] or industrial sensing [72], or in the present case flash flood detection. Floods are one of the most commonly occurring natural disasters [14], accounting for more than half of natural disasters worldwide. They have caused more than 120,000 fatalities in the world between 1991 and 2005 [15], and are a major problem in many areas of the world. While most floods occur outside of urban areas, the recent trend towards urbanization will likely make urban floods more catastrophic due to the concentration of population into relatively small urban areas.

Among floods, flash floods are short fuse weather events, that last less than six hours. Most flood fatalities are in fact caused by flash floods, and most flood victims die because of drowning [16]. This could be avoided by providing accurate flash flood maps to the population in real time. Unfortunately, at the present time little warning exists beyond weather forecasts, which are nonspecific (lack of exact location of the flood, severity of the flood, temporal evolution) and not reliable (*i.e.* these warnings are associated with a relatively high false alarm rate).

Monitoring floods in real time somehow requires sensing the flooding conditions [18]. Fixed water level sensors are only adapted to river monitoring, and instrumenting entire hydrological basins, which can cover hundreds of square kilometers, is economically infeasible. Satellites are similarly unable to monitor water levels and flows remotely: optical measurements are impossible during floods, and the vertical resolution of current synthetic aperture radars (tens of centimeters) is insufficient for the task.

Existing work ([73, 74, 75, 76]) relies on either contact-based sensors, or non-contact camera-based sensors which are unable to provide a direct water level measurements (such sensors can only provide a binary information: presence of water or not). In [77], the authors use ultrasonic rangefinders to monitor floods, but do not consider the environmental perturbations to the measurement, which is the focus of the present work. Such perturbations severely affect the accuracy of the sensor, and can lead to false or missed detections.

In this Thesis work, we propose a new type of flash flood sensor combining ultrasonic rangefinders with passive infrared temperature sensors. This sensor can be used as a backbone for an urban flash flood wireless sensor network architecture, since it can monitor pluviometry, water presence and water level with a relatively high accuracy. While the measurement of distances using a calibrated ultrasonic rangefinder is easy, when environmental conditions are well known, the present problem involves the estimation of a distance from time-of-flight measurements and from a model of the atmospheric layer between the sensor and the ground. Since this model is encoded by a Partial Differential Equation (PDE) which has numerous uncertain parameters, we choose a non-model based approach to compute the water levels directly, from raw temperature and distance measurements.

The emphasis of the present work is on water level detection (using the proposed sensor [78]), though this sensor can also be simultaneously used for traffic flow monitoring (by monitoring the temperature and distance disturbances created by vehicles passing by the sensor), or rain rate detection, making it a cost-effective solution. We show that simple temperature correction methods fail to provide a sufficiently accurate measurement, and that machine learning or nonlinear dynamical model-based regression methods can provide a solution to the sensing problem.

In particular, Artificial Neural Networks (ANN) have an excellent accuracy, and have a low computational complexity (for the chosen number of neurons and layers) which makes it suitable to low-power embedded platforms.

The following list summarizes the contributions of the present work for the flood monitoring part of the thesis over existing literature:

- Novel, dual use (traffic and flash flood) sensor system that addresses a key economic requirement in flash flood sensor networks (since flash floods happen very infrequently).

- New preprocessing scheme based on L_1 regularization for real-time sensor fault detection and corresponding missing data inference.

- Use of multiple machine learning techniques, ranging from artificial neural networks and fuzzy logic to nonlinear regression on preprocessed sensor measurement data to estimate water levels and learn the proper compensation to apply over all environmental conditions.

- Implementation of the corresponding algorithms on a low-power experimental hardware platform.

- Extensive validation over a period of one year, including the successful detection of the only flash flood event occurring over the period, without any false detection.

The rest of this chapter is organized as follows. Section 4.1 describes the dual ultrasonic/passive infrared sensor network used in this study. Section 4.2 describes the nature of the urban flood detection problem and the research questions addressed in this paper, going through the possible models that can be used to estimate water levels using raw measurement data, (include linear models (ARMAX)) and how these models can not help in this situation and then in the same section, we emphasize on the need of a dedicated preprocessing stage. We then show in section 4.3 our proposed solution (using our sensor nodes described in section 4.1) and its performance comparing it to the existing models using norm two (L_2) and norm infinity (L_∞) approaches, as well as, investigating the spatio-tempral robustness of the proposed solution model parameters. Section 4.3.7 investigates the detection of an actual minor flooding incident using the proposed approach.

We then describe a computational platform that can solve the proposed neural network training and prediction problems in real time in section 4.4.

4.1 Sensing principle

4.1.1 Sensor design considerations

Sensing flash floods in cities is challenging, since the sensors must have an extended lifetime, measure the water levels in all flow conditions, and be capable of self-monitoring (to make sure they are always functional).

Sensors have been investigated in the past for flood monitoring applications [79], in particular ultrasonic water level measurements on bridges [80], [81] or pressure sensors for water level measurements of rivers [1]. In the present case, the constraints detailed above prevent contact sensors such as pressure transducers [82] from being used. Indeed, these sensors would be unable to measure static and dynamic pressures independently, and their measurements would be affected by their orientation with respect to the water flow. They could also be affected by debris or rocks carried by flash floods, and would need to be periodically tested in water.

Among non-contact sensing technologies, three main technologies could be thought of: ultrasonic rangefinders and Ultra Wide Band (UWB) radars and LIDARs. Ultrasonic rangefinder are much cheaper and more accurate than both UWB and currently available LIDARs, though they can be affected by environmental parameters, such as temperature or humidity. In this work, we choose ultrasonic rangefinders for their very low cost.

4.1.2 Sensor description

The flood sensor node that we investigate in the research work [83] contains six passive infrared (PIR) sensors and one ultrasonic (US) rangefinder connected to a microcontroller platform that we have developed for this specific purpose. The six PIR sensors (which are also used for traffic monitoring) are Melexis MLX90614 connected to the platform via SMBus. These sensors measure both the ground temperature in their field of view and their actual temperature. The ultrasonic rangefinder is a MaxBotix MB7066 measuring its distance to objects below it and sending these measurements to a microcontroller via a serial port. The microcontroller platform is developed by our group [84], and is based on an ARM Cortex M4 processor operating at 32 MHz.

At this stage, the microcontroller platform applies a median filter to the raw distance and temperature measurements provided by all sensors. The time window of this median filter is currently 30s, and gives a reliable estimate of the actual ground and sensor temperatures, as well as raw ground distance. Median filtering was chosen over moving average filtering because of the presence of vehicular traffic, which creates perturbations in temperature and distance measurements. When traffic is relatively light (which is always the case on the deployment sites), these perturbations are completely cancelled by the median filter. The measurements are sent wirelessly to a sink node, and are then pushed to an input database. The complete system is represented in Figure 4.1.

The complete sensor structure is made of a lightweight aluminum alloy and weights less than 6 kg. The structure has been designed by our group using CAD tools, and

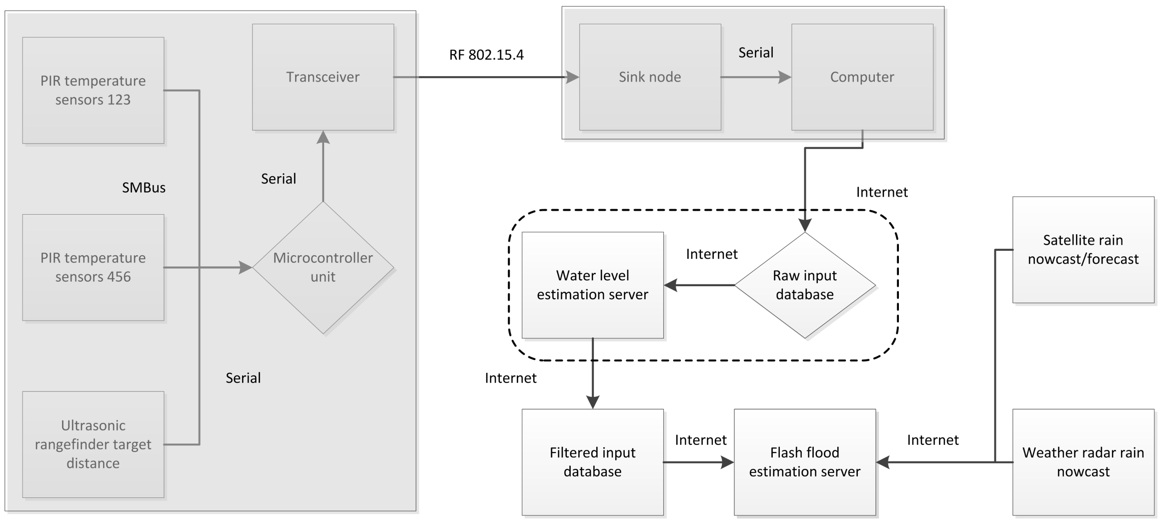

PIR temperature sensors 123
Transceiver
RF 802.15.4
Sink node
Serial
Computer
SMBus
Serial
Internet
PIR temperature sensors 456
Microcontroller unit
Internet
Water level estimation server
Raw input database
Satellite rain nowcast/forecast
Serial
Internet
Ultrasonic rangefinder target distance
Filtered input database
Internet
Flash flood estimation server
Internet
Weather radar rain nowcast

Figure 4.1: System representation. The sensor described in this work corresponds to the leftmost part of this diagram.

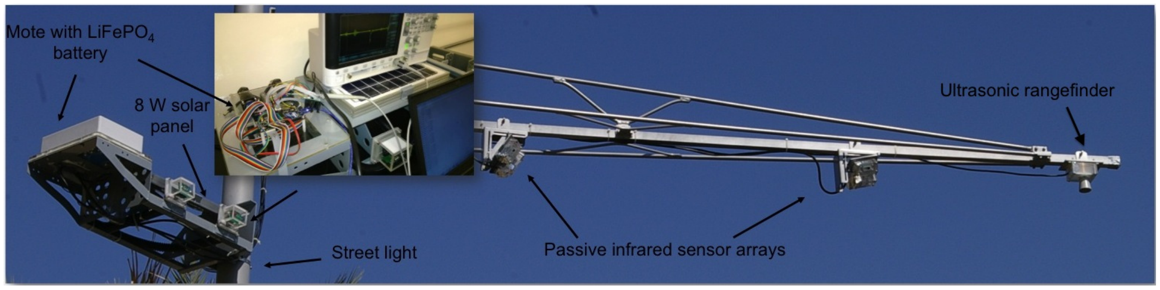


Figure 4.2: Flood sensor node installed on a public street light in KAUST university campus

manufactured by a subcontractor. Four of these sensors have been deployed on street lights within two residential areas located 120km apart for the dual use of traffic and flash flood monitoring. These sensors are operational since November 29, 2013, and are illustrated in Figure 4.2.

4.2 Problem definition

The temperature and distance to the ground measurements generated by the sensor over 6 days (under normal conditions, so that fluctuations are only due to temperature change) are shown in Figure 4.3. As can be seen from this figure, the raw distance measurements vary significantly over the period (about 12 cm) despite the fact that no flooding occurred over this period. These variations in observed distance are caused by the dependency of the speed of sound on air temperature. For most distance measurement applications, the air temperature effects cause an error that is usually negligible, specially when the temperature of the air is uniform in space. For our sensor, installed at a height of 5 meters (to clear the below traffic) and at usual temperatures, a $1°C$ increase in temperature causes a 1 cm increase in the measured distance. In addition, the temperature of the air is far from being uniform, with ground temperatures up to $20°C$ higher than air temperatures. Since the distance measurement errors caused by temperature variations are larger than the expected precision of the sensor, we need to somehow estimate the true distance to the ground using the raw distance and temperature measurement data.

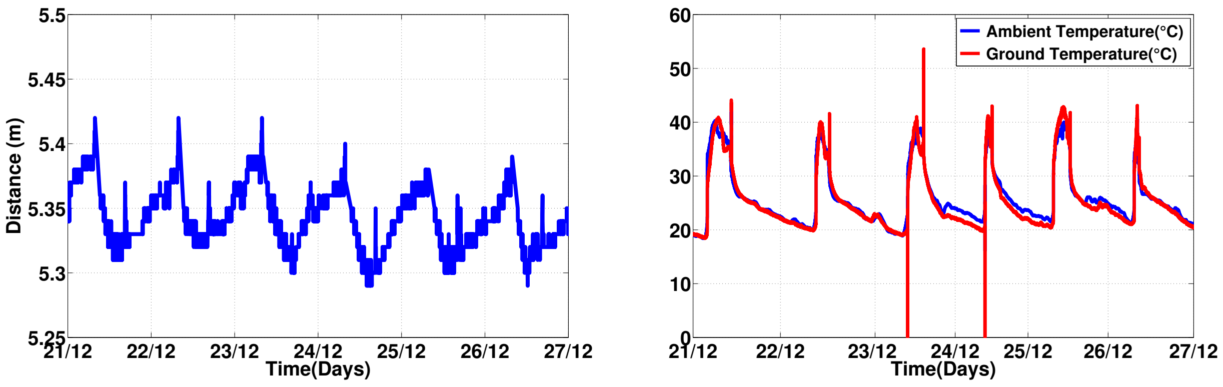


Figure 4.3: Left: Raw distance measurements from ultrasound sensor. Right: ambient and ground temperature measurements

This problem can be formulated as follows. The ultrasonic rangefinder measures a time-of-flight (which is converted into an estimated distance by assuming a fixed speed of sound). The time of flight $F(t)$ of the ultrasonic wave (measured at a given time t) can be computed as follows [85]:

$$F(t) = 2 \int_0^{z_0} \frac{dz}{c(\theta(z,t))} \tag{4.1}$$

where $\theta(z)$ is the air temperature at altitude z. One of the main difficulties arising in this problem is the estimation of the function $\theta(z,t)$. This function depends on multiple factors that are not directly measured by the sensor, including for example cloud coverage, presence of shadow from buildings, heat island effects and ground albedo. To increase the accuracy of the sensor, we need to estimate the correction to apply to the raw distance measurements of the ultrasonic sensor. The correction factor represents all the perturbations caused by the uneven temperature profile in the air layer below it. For the reasons mentioned above, it is infeasible in practice to model the air layer temperature using a PDE, since the boundary conditions and the model parameters of the problem are unknown. We thus investigate several model and non-model based approaches to correct the measurements, ranging from linear ARMAX models to ANNs.

In this context, our problem is formalized as follows. Let us denote, by $e(t)$, the difference between the measured (*i.e.* the raw output of the ultrasonic rangefinder) and the actual ground distance, which is known when no flash flood occurs. Our objective is to estimate $e(t)$ given $T_a(\cdot)$ and $T_g(\cdot)$, where $T_a(\cdot)$ and $T_g(\cdot)$ are the sensor measurements of the ambient and ground temperatures respectively.

4.2.1 Naive temperature correction

We first investigate the need for a complex temperature compensation model [86] by checking if simpler naive correction methods can apply. We know that the measured time of flight is given by equation 4.1, in which $\theta(z,t)$ is unknown. Let us first assume that $\theta(\cdot,t)$ is constant, that is, the temperature in the air column is uniform. With this assumption, equation 4.1, becomes:

$$F(t) = 2\frac{L}{c(\theta(t))} \tag{4.2}$$

where $\theta(t)$ is the uniform temperature in the air column. For the two applications below, we used $c(\theta(t)) = 331.3 + 0.6 \cdot \theta(t)$ in m/s, which represents the approximate speed of sound in function of the air temperature $\theta(t)$ in $°C$, on a narrow temperature range around $20°C$.

In Figure 4.4, we assume that $\theta(t) = T_a(t)$, *i.e.,* the air temperature in the column is identical to the ambient temperature measured by the infrared sensor. This assumes that the heating effects of the sun on the sensors are negligible (that is, the sensor is in thermal equilibrium with the ambient air). This occurs in practice when the wind is very high (which increases the heat transfer between the sensor and the air), or when the solar forcing is negligible (for instance during the night, or during overcast conditions).

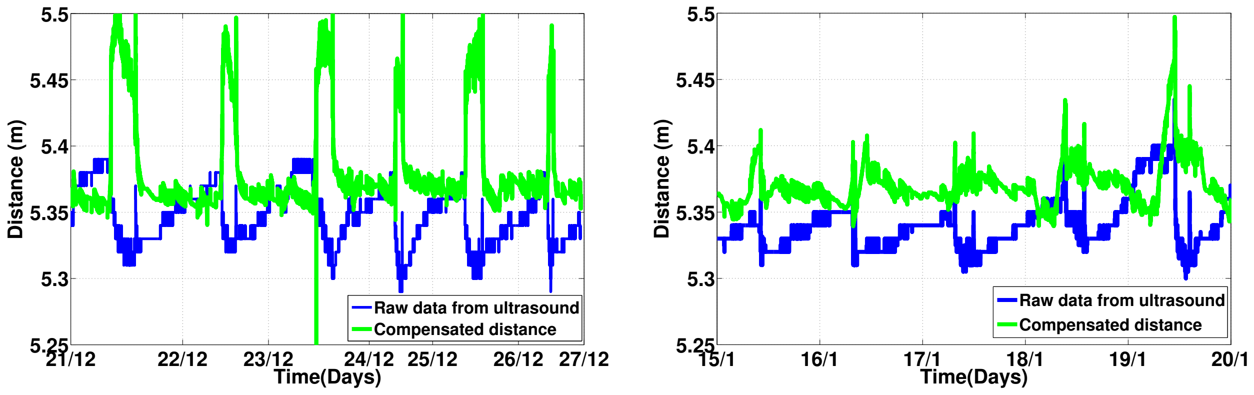


Figure 4.4: Left: Predicted distance measurement computed using ambient temperature sensor measurements, and Right: weather station air temperature measurements.

As can be seen from Figure 4.4, the estimation during the night is very good, though the compensation fails during the day. This is caused both by the assumption that the air temperature is uniform (which is wrong during the day), and by the fact that the ambient temperature measured by the sensor can be higher than the air temperature due to solar forcing. To remove the solar forcing effect on the ambient temperature measurement of the PIR sensor, we used calibrated air temperature measurements from the closest weather station. The corresponding predicted distance measurements are illustrated in Figure 4.4, and similarly show large variations with time, whereas the actual distance between the sensor and the ground is constant. This shows that the air temperature cannot be assumed to be uniform.

Another simple correction method could be to use the average speed of sound in the air layer, assuming that the air temperature profile $\theta(z,t)$ varies linearly between the ground and the sensor, that is:

$$\theta(z,t) = T_g(t) + \frac{T_a(t) - T_g(t)}{L} z$$

where L is the distance between the sensor and the ground, and z is the altitude above the ground level. This method does not improve the accuracy of water level estimate beyond the naive correction methods introduced earlier. This analysis shows that more advanced methods are needed to estimate the distance between the sensor and the ground accurately.

These methods have to rely not only on current temperature measurements, but also on the history of measurements, resulting in a dynamical system. In this part of the thesis, we investigate various models to compensate the thermal effects and correct distance measurements generated by the sensor. One of the simplest of such models involves a linear dynamical system with inputs, otherwise known as ARMAX (Auto Regressive Moving Average with Exogenous inputs) models, which we now investigate.

4.2.2 Auto-regressive moving average exogenous (ARMAX) fitting

ARMAX can be used for modeling time series and predicting future observations by combining a number of previous observations (Autoregressive), a number of noise terms (moving-average), and a number of external (exogenous) inputs, which are $T_a(\cdot)$ and $T_g(\cdot)$ in the present case. Formally, the error $e(t)$ between the measured distance and the actual distance is given by:

$$\begin{aligned} e(t) = \sum_{i=1}^{q} \varphi_i e(t-i) + \sum_{i=1}^{q} \theta_i \varepsilon(t-i) + \\ \sum_{i=1}^{q} \eta_i T_a(t-i) + \sum_{i=1}^{q} \xi_i T_g(t-i) + \varepsilon(t) \end{aligned} \tag{4.3}$$

where $\varepsilon(t)$ is the white noise, φ_i, θ_i, η_i and ξ_i are the model parameters. q is the number of terms considered in autoregressive, in moving-average and in exogenous inputs, or the order of ARMAX model. In our study, we set the order to a limited number of coefficients (20) in order to avoid over-fitting, and to allow better comparison with other methods.

4.2.3 Supervised learning

Supervised learning has been extensively studied in machine learning and has been successfully applied to a wide range of applications. Given the inputs and outputs of a system, supervised learning algorithms model the functional relationship between inputs and outputs, with no need of knowing the system working principles. In other words, it can predict how the system reacts on the given inputs without knowing how a system works. In our study, the effects of the two temperature measurements provided by the sensors on the measured distance are too difficult to model accurately. Nevertheless, this complex relationship can be modeled through supervised learning, with inputs of the ambient and ground temperatures and outputs of the observed raw distance measurements from ultrasound sensors and the actual distance. Note that when floods do not occur (*i.e.* most of the time), the actual distance is perfectly known, and the output of the system is the raw distance measurement generated by the ultrasonic rangefinder. Once the relationship is learned, the actual distance between the sensor and the ground can always be estimated given the ambient and ground temperature measurements. Therefore, supervised learning can be viewed as a natural solution to our water level prediction problem.

Non-Linear regression:

The first supervised learning method we used is non-linear regression, which models the target signal by a non-linear function. In this paper, the following quadratic function is utilized:

$$e(t) = \sum_i (b_{3i+1} T_a(t - k \cdot i) - b_{3i+2} T_g(t - k \cdot i) + b_{3i+3})^2$$

where $e(t)$ represents, as before, the difference between measured distance and actual distance at time t. $T_a(\cdot)$ and $T_g(\cdot)$ represent the ambient (sensor) and ground temperatures respectively, and $b.$ are the parameters of the non-linear regression function. The combination $(k \cdot i)$ represents the discrete time horizon used for the prediction. To avoid over-fitting, we chose $k = 5$, which results in a 15-parameter model taking into account the five most recent pieces of data. This number of parameters used in this model (15) is comparable to the 20 parameters used in the ARMAX analysis detailed above.

This quadratic regression function is selected after empirical comparison of models with different parameters. It is the best performing model that less suffers the over-fitting problem and is more accurate on prediction. Also, the proposed model allows us to capture the influence of the two measured temperatures (ambient and ground temperatures) as well as the coupling (caused by the effects of solar irradiance, the heat island effect and the presence of shadows on the ground) between them. The fitting is done through successive approximations, for which we limit the number of iterations to 100 and the tolerance to 10^{-8}. Robust fitting options can be added to improve the root mean square error performance of the predicted values.

Neural networks:

Artificial Neural Networks (ANN) are a computational model inspired by the central nervous systems of animals [87]. This concept and a few possible applications in industrial electronics are summarized in [88], [89]. In this specific field, there already exist a large number of applications of neural networks, some of which include motor drives [90] or power distribution problems dealing with harmonic distortion. Due to their nonlinear nature, they have also become an integral part of the field of control systems engineering [91], [92].

Neural network architecture: Neural networks can be considered as a combination of a set of nonlinear functions that are organized structurally layer by layer. They are thus also called MLPs (Multilayer perceptron). They are well suited to various application problems due to their "universal approximation" property: any continuous function can be uniformly approximated to an arbitrary accuracy by neural networks, given enough hidden units with any of a wide variety of continuous nonlinear hidden-layer activation functions, see for instance [93], [94], and [95].

In our problem setting, there are two inputs (ambient and ground temperatures), and one output (the raw distance measurement from the ultrasound). Each neuron in the hidden layer connecting to the inputs is fed with the input variables $x_1, ..., x_D$. The complete network function is obtained by:

$$y_k(x, \mathrm{w}) = \alpha(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_{j0}^{(1)}) + b_{k0}^{(2)}) \tag{4.4}$$

where $j = 1, ..., M$ (M is the total number of hidden neurons in the this layer). The parameters $w_{ji}^{(1)}$ and $b_{j0}^{(1)}$ ($w_{kj}^{(2)}$ and $b_{k0}^{(2)}$) are the weights and the biases for the inputs to the first (second) hidden layer. $h(.)$ is an activation function (in the present case the activation functions are sigmoid functions). The output is a function $\alpha(.)$ (which will be discussed in the following section) of the second layer variables. Thus the neural network model can be considered as a nonlinear function mapping a set of input variables $\{x_i\}$ to a set of output variables $\{y_k\}$ controlled by a vector w of adjustable parameters.

Levenberg-Marquartdt back-propagation training function: Training a neural network involves the tuning of the weights and biases of the network. The objective is to maximize the network prediction performance, which corresponds to minimizing the difference between all network outputs y_k and desired outputs or targets t_k on validation data. The computational time required for the training algorithm depends on many factors, including the complexity and objective function of the problem, the size of training data, the number of parameters in the network, and whether the network is being used for pattern recognition (discriminant analysis) or function approximation (regression [94]). For our particular problem, we are interested in the function approximation problem with a few hundred weights in a moderate size network. In this specific case, the Levenberg-Marquardt algorithm has been proven to have the fastest convergence [96]. It updates the network weights and biases in the direction in which the performance function decreases most rapidly. This advantage is more noticeable when implementing NN on the proposed platform of this work, since sensors have a limited bandwidth, and that transmitting all the data would be of a huge energy consumption, thus it is very critical to consider efficient online training.

Having stated all of this and trying different modules we realized that we need an efficient preprocessing tool in addition to the online training. Indeed, the raw data generated by the infrared temperature sensors and the ultrasound rangefinder have different scales, and are sometimes exhibiting inconsistencies, as can be seen from Figure 4.3 which is a major game changer in our application. Therefore, an effective preprocessing procedure is essential in accurately sensing floods based on measurement data produced from sensor nodes.

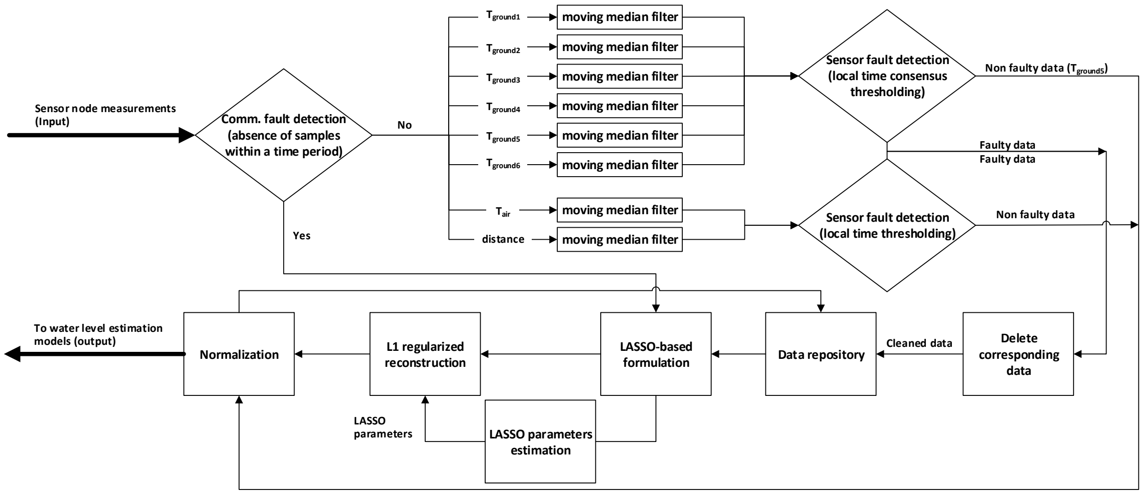

Sensor node measurements (Input)
Comm. fault detection (absence of samples within a time period)
No
Yes
T ground1
T ground2
T ground3
T ground4
T ground5
T ground6
T air
distance
moving median filter
Sensor fault detection (local time consensus thresholding)
Non faulty data (T grounds)
Faulty data
Faulty data
Sensor fault detection (local time thresholding)
Non faulty data
To water level estimation models (output)
Normalization
L1 regularized reconstruction
LASSO-based formulation
Data repository
Cleaned data
Delete corresponding data
LASSO parameters
LASSO parameters estimation

Figure 4.5: The preprocessing of the measured data from the sensor node. The processed data are served to the estimation models.

4.3 Proposed Solution and System performance

4.3.1 Preprocessing of measurement data

As stated previously in section 4.2, the raw data generated by the infrared temperature sensors and the ultrasound rangefinder have different scales, and are sometimes exhibiting inconsistencies, as can be seen from Figure 4.3. Therefore, an effective preprocessing procedure is essential in accurately sensing floods based on measurement data produced from sensor nodes. Our proposed preprocessing procedure is illustrated in Figure 4.5. It mainly involves two main processes: fault detection and missing data reconstruction.

Sensor fault detection:

Sensor faults can be caused by multiple factors. In the present case, these faults were mainly due to gateway failures (due to a loss of the internet connection), and brownouts of the sensor caused by faulty charging circuits in a narrow solar power input range when collecting the measurement data.

We first check the network faults (communication faults), which are identified by detecting time periods that have more than 15 minutes (*i.e.* 90 samples at the 0.1Hz message update rate) without reception of data from sensors. When such faults are detected, the missing data in the blank periods are reconstructed directly by the LASSO-based formulation which will be explain in the following section. To remove outliers (caused by vehicles passing below the sensor), we apply a moving median filter to the temperature and distance data. Finally, we detect sensor fault by analyzing the consensus among all six infrared temperature sensors in the node. Whereas for the ambient temperature and the distance measurements the consensus is not applicable since we have one of each in a sensor node. We then use a sliding window, in which we calculate the mean μ and the standard deviation σ of the measurement data of all the sensors . The records that deviate more than 3σ from the mean are considered anomalies, and excluded from the dataset. Since the main cause of sensor noise is thermal noise [97], the sensor measurements follow a normal distribution, given that the thermal anomalies caused by the vehicles are suppressed by the median filter. Hence, 99.7% of the values fall within the $[\mu - 3\sigma, \mu + 3\sigma]$ window. Most of the data excluded using this approach is indeed faulty data. All data gone through the fault detection process are stored and served to the next step, approximating the missing data to complete the measurements.

$\underline{L_1 \text{ regularized reconstruction}}$:

In order to reconstruct the missing data in real-time, we leverage the data generated during the previous days as follows. Let us consider the time series $s_i(\cdot)$, obtained during the previous $i \in \{0, ..., i_{\max}\}$ days, and let $j \in \{0, ..., j_{\max}\}$ represent a time index in a day (in practice, j ranges from 0 to 8640 since we have a sampling rate of $0.1Hz$). We assume that the current measurements $s(j)$ can be written as a linear combination of the past measurments, as follows:

$$s_{\text{estimate}}(j) = \sum_{i=0}^{i_{\max}} \alpha_i s_i(j) \tag{4.5}$$

where $\alpha_1, \cdots, \alpha_{i_{\max}}$ are model parameters. Finding these parameters can be done by minimizing the estimation error, for example in the least squares sense:

$$\min_{\alpha} \sum_{j \in J} \left(\sum_{i=1}^{i_{\max}} \alpha_i s_i(j) - s_0(j) \right)^2 \tag{4.6}$$

where $J \subset \{0, ..., j_{\max}\}$ is the set of times (for the current day) associated with non-faulty measurements. s_0 is the current day measurements.

In general, $|J|$ is considerably larger than $i_{\max}$, and the problem is overdetermined. The main issue with this formulation is the fact that any given day should only have a few identifiable features from the previous days patterns, and thus, α should be sparse. To enforce this sparsity constraints, we add a L_1 norm regularization term to the above formulation, leading to the formulation of our estimation problem as a LASSO (least absolute shrinkage and selection operator) [98]:

$$\min_{\alpha} \sum_{j \in J} \left(\sum_{i=1}^{i_{\max}} \alpha_i s_i(j) - s_0(j) \right)^2 + \lambda ||\alpha||_1 \tag{4.7}$$

where λ is the regularization parameter that trades off fitting quality and sparsity. Sparsity also improves the robustness of the regression to noise.

Problem (4.7) is a quadratic program (QP) which can be solved using standard convex optimization methods. The results of the preprocessing scheme is illustrated in Figure 4.6. In this Figure, we show the raw measurement data, along with the output of the fault detection stage, and the output of the LASSO-based data reconstruction stage. We applied the above algorithms to filter measurement data over extended periods of time. Figure 4.7 illustrates the reconstructed measurement data between December 4th 2013 and December 21st 2013, which is used as training data for the artificial neural network-based water level estimation scheme, introduced in the subsequent sections.

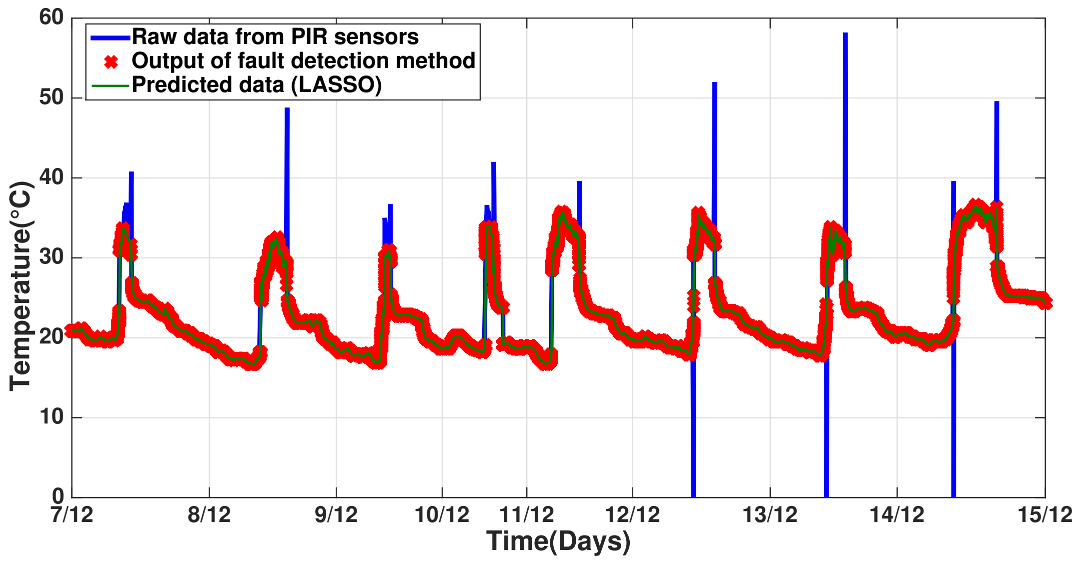


Figure 4.6: Comparison between raw PIR sensor data (blue) and reconstructed PIR data (green) at the end of the preprocessing stage for a sample of the data in December (without normalization).

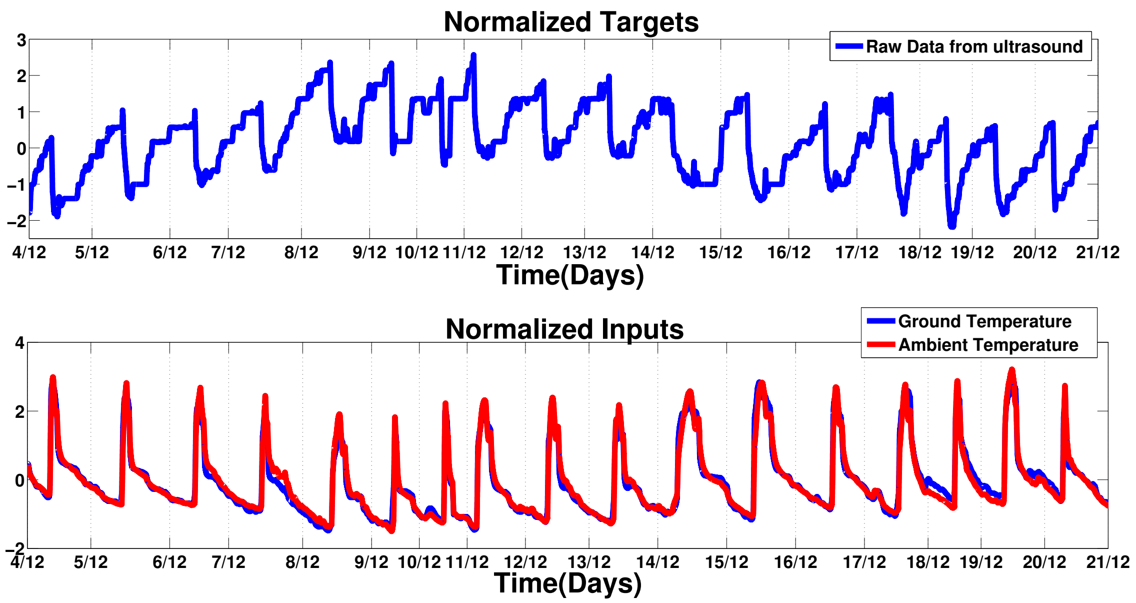


Figure 4.7: Top: Normalized and filtered raw distance measurements (for the preprocessed training dataset). Bottom: Normalized and filtered ambient and ground temperature measurements (for the preprocessed training dataset).

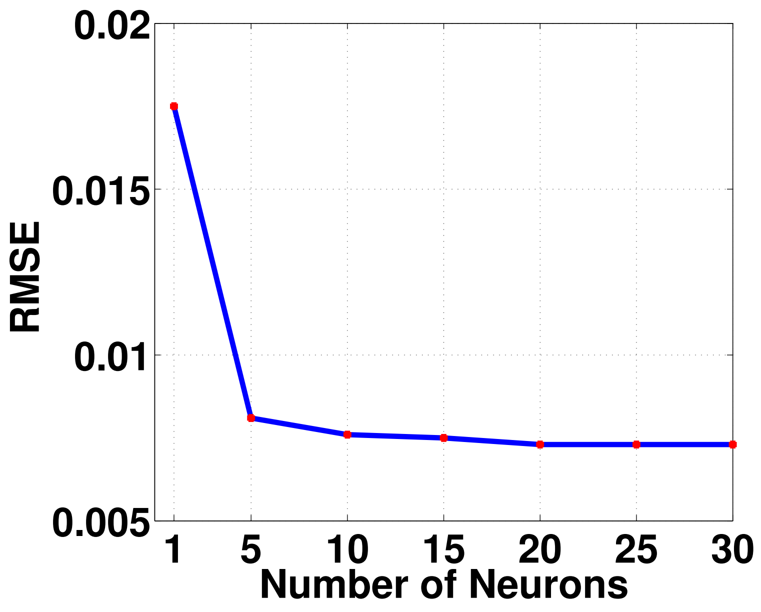


Figure 4.8: Relation between the number of neurons in the hidden layer of an ANN and the root mean square value (RMSE). As can be seen from this Figure, the RMSE only improves marginally as we increase the number of neurons past 5 neurons.

This section reports the performance of different models we studied in the previous section. The data set used in this validation has been generated from the sensors deployed on our university's campus.

4.3.2 Artificial neural networks performance

The number of neurons has been set to achieve good performance on the validation dataset, considering the trade off between the computational time required to train the network and the performance of root mean square error (RMSE) that can be reached. Furthermore, we had to fix the number of neurons to be comparable with the other methods we used to estimate the water level (since the number of neurons determines the number of coefficients to be optimized). RMSE only improves marginally as we increase the number of neurons past 5 neurons, as can be seen from Figure 4.8.

We implemented the neural network model using the neural network toolbox of

`Matlab`, running on a `Mackbook i7`. We were able to evaluate and train the neural network using the Levenberg-Marquardt algorithm with one hidden layer of 10 neurons in a two inputs and one output setting.

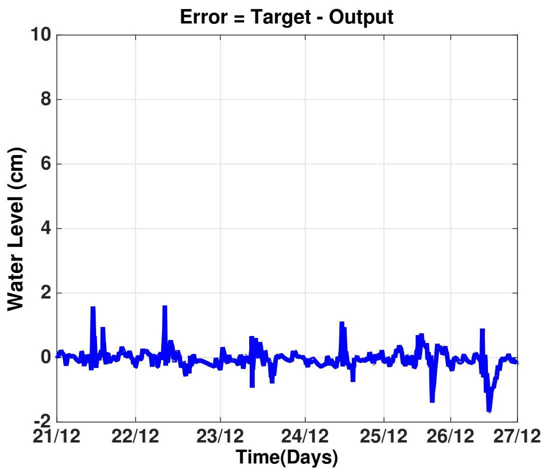
The dataset consists of 205000 data points, as illustrated in Figure 4.7. We arbitrarily divided it into a training dataset, a validation dataset and a testing dataset. The training dataset corresponds to 70% of the overall data, while the validation and testing datasets correspond to 15% of the data each. The validation dataset is used during the training process, to halt training when the performance of the estimation on the validation dataset stops improving.

Figure 4.9 shows the histogram of the error between output and targets for the training, validation and testing samples. It can be seen in the figure that the histogram fits a normal distribution with a maximum detected error on validation and on testing samples of less than 2 cm.

The absolute difference (or called Mean Absolute Error, `MAE`) between predicted and actual distance measurements is shown in Figure 4.9. This difference is also considered as the prediction of water level. Since during this period (December 21st to 27th) no flood occurred, the actual water level was always 0 cm. As can be seen from the figure, the deviation of prediction from 0 is less than 1.85 cm. This low prediction error shows the good performance of a neural network on water level prediction.

4.3.3 Comparison of prediction models

Table 4.1 shows the comparison between all the approaches investigated in this work for water level estimation, on an identical dataset (i.e., similar range of validation dataset from December 21st to 27th, 2013, and when training is required we used similar inputs, preprocessed data, shown in Figure 4.7). Their performance is compared in terms of prediction accuracy (measured by the maximum absolute error



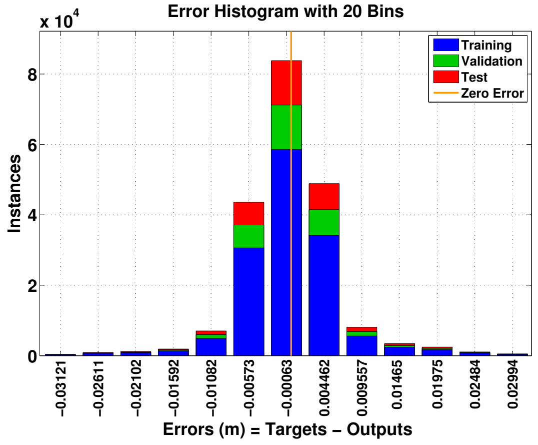


Figure 4.9: Left: Water level prediction during a week of December 2013 for the sensor node "A77D". Since no flood occurred during this period, the actual water level was always 0 cm. The deviation of predicted value from 0 can be considered as prediction error, which is less than 1.85 cm. Right: error distribution over the training, testing and validation datasets.

(L_∞) and RMSE), efficiency (computational time in both training and prediction) and prediction stability (standard deviation of errors in 5 independent runnings of the same testing dataset).

Besides naive compensation and ARMAX, our proposed neural network approach is compared to three other machine learning techniques, non-linear regression, decision tree and fuzzy logic, which are widely used supervised learning methods for making predictions [99]. The neural network and non-linear regression are introduced in section 4.2.3. We designed a fuzzy logic predictor using two membership functions on our two inputs, and thus resulting in four fuzzy rules. Then we use a hybrid learning algorithm to tune the parameters of a Sugeno-type fuzzy interference system [100] which is a combination of the least-squares and back-propagation gradient descent methods to model a training data set. For comparison purposes, we ensured that the number of parameters to optimize is similar to the other models and the same training and testing data are used for evaluation. Decision tree for regression is employed for estimating water level at leaf nodes, while each branching node splits values of one input variable. The tree is trained by the ”Regression Tree” functions provided in `Matlab`, with the same training data we used for the other models.

On accuracy, the four machine learning (ML) methods presented in this work are performing better than the naive compensation and ARMAX, in which the neural network (NN) approach performs the best. With regard to computational time, the naive compensation is associated with a zero training time, while the training time for the ARMAX estimation method is negligible (since the model is linear). Each of the ML methods involves a training process, and thus demands some time for learning. However, it is more important to compare the computational time in prediction, because ML training can always be conducted offline. Except for ARMAX, the five approaches need around $5-6$ seconds to make the prediction for the complete testing dataset (i.e. 30% of the complete dataset of 205000 data points), that is, 8.76×10^{-5} seconds to make one prediction, which is fast enough to satisfy the real time requirements.

The ML methods are also more stable than others, and thus can guarantee that the prediction is reliable.

Table 4.1: Comparison of the performance of 6 models

Model	L_∞(cm)	RMSE	Computing time (sec.)		Error Std.
			Training	Prediction	
Naive Compensation	15.3	0.0547	0	5.3	0.0504
ARMAX	4.9	0.0164	0	272.3	0.0152
Fuzzy Logic (ML)	2.59	0.0413	168.6	5.5	0.0377
NL Regression (ML)	2.57	0.0082	28	5.7	0.0083
Decision Trees (ML)	2.15	0.0158	26.6	5.9	0.0158
NN (ML)	0.6	0.0058	975	5.2	0.006

4.3.4 Minimization of errors in the L_∞ sense

Most ML techniques involve the minimization of the prediction error in the L_2 norm sense, as we investigated above. Since flash floods are extremely rare events, we are also interested in making prediction with minimized error in the L_∞ sense, to avoid false alarms in the flood monitoring system. The infinity norm is better than norm

2 for the task, since we want to detect events based on thresholding, and a very low norm 2 (but high norm infinity) in the training (non flood) data is unacceptable, since it would lead to false detections.

Our objective in this subsection is thus technically to minimize the prediction error in the L_∞ sense instead of the L_2 norm sense, though the L_∞ error usually improves when the L_2 error is reduced.

Minimizing the prediction error in the L_∞ sense is straightforward in the case of a linear model: it can be formally written as:

$$\min_{W} \quad \|XW - Y\|_\infty$$

where X is the input, W is the model coefficient to be optimized, and Y is the output. This resulted Linear Program (LP) problem is solved by the `SDPT3` package working under `Matlab` in this paper.

We investigate both the linear model (ARMAX) and the nonlinear model (non-linear regression) for minimizing error in L_∞. Table 4.2 shows the performance of ARMAX and non-linear regression, measured in maximum absolute difference (L_∞), RMSE and standard deviation of errors (in 5 independent runnings of the same testing dataset). Comparing the L_∞ values of these two methods in Table 4.2 and Table 4.1, we can see that they have smaller L_∞ when minimizing L_∞ error than when minimizing L_2 error. The non-linear regression model with minimized L_∞ error is comparable to the ANN model with minimized L_2 error, which is the best as observed in Table 4.1.

With regard to the RMSE, both ARMAX and non-linear regression have high error value when minimizing error in L_∞. This is understandable, as they by nature lead to small RMSE (a performance metric defined in L_2) when minimizing error in L_2.

Table 4.2: Comparison between ARMAX and NL regression when minimizing error in L_∞

Model	L_∞(cm)	RMSE	Error Std.
ARMAX	2.28	1.2593	1.0128
NL Regression (ML)	0.55	0.3218	0.2572

4.3.5 Temporal robustness of ANN model parameters

In this section, we evaluate the time dependence of the training parameters. For this, we train the artificial neural network model using data from the early December (December 4th to 21st), and validate the performance on three different sets of data (timely separated), namely late December (December 21st to 27th), January and February.

Table 4.3 shows the performance of the estimation models of the neural networks on the sensor node "A77D" using training data from early December 2013 only. As can be noted, the prediction performance slightly degrades in February. However, the estimation error is acceptable (within about 2 cm) for flash flood monitoring applications. This suggests that the coefficient can be used for large durations (e.g., tens of days) before degrading (hence it needs to be tuned), which enables large datasets to be integrated in the training process.

Table 4.3: Evaluation of temporal robustness: comparing prediction accuracy in different months when using NN trained by data in early December, 2013.

Month	L_∞(cm)	RMSE	Error Std.
Late December	1.63	0.0215	0.0213
January	2.08	0.0267	0.0267
February	2.15	0.0272	0.0272

4.3.6 Spatial robustness analysis of ANN model

To evaluate the robustness of the model parameters with respect to spatial changes, we installed two sensor nodes in a close proximity (during January 2014): sensors "A77D" and "8F90". Node "8F90" is placed in an area that can be shadowed by a

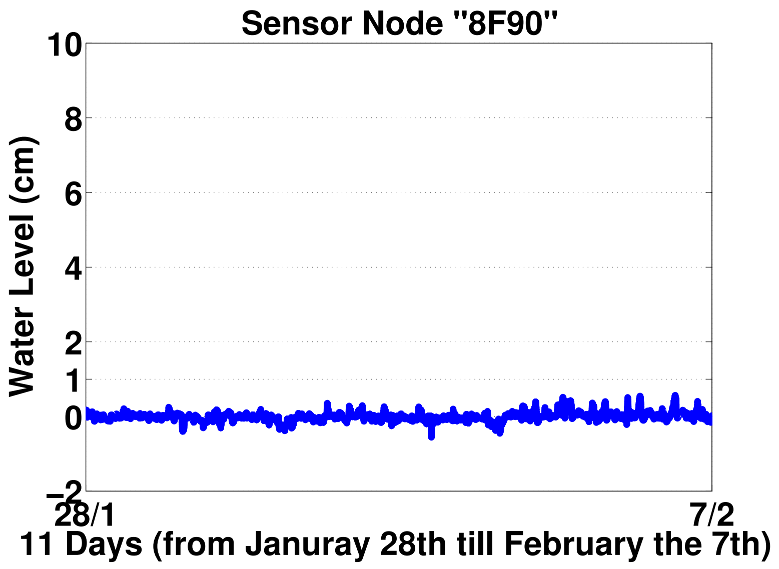


Figure 4.10: Estimated water level at the sensor node ”8F90” from January 28 to February 7, 2014. In this test, the artificial neural network is trained onboard by the sensor node itself.

building at some times of the day, unlike node ”A77D”.

In a similar manner to the analysis done to ”A77D”, we estimated the water level and the norm infinity error of estimation during a period of 11 days is shown in Figure 4.10 (picked right after we installed the node early in January 2014). It can be seen from Figure 4.10 that the error is less than 0.6 cm.

Then, in order to evaluate the spatial robustness, the neural network is trained by data from sensor ”A77D” during December 4th and 21st, 2013, to predict the data for the sensor node ”8F90” over the same testing period of the 11 days.

As expected, the performance of the prediction of ”8F90” slightly degrades in this case: RMS error of 0.275 vs. 0.0058 and norm infinity error of 1.735 cm vs. 0.6 cm for the training with the data of ”8F90”, compared to the training with the data of ”A77D”. This acceptable error (i.e. less than 2 cm) allows us in practice to use training data from the nearest sensors to immediately generate distance measurement data from a newly installed sensor, without having to wait for a training period.

4.3.7 Validation on an actual flooding incident

Flood incidents are usually rare, but one such incidents occurred in the test period, the only incident which has occurred to date. This incident has been detected by two sensors installed in Umm Al Qura University in Mecca, Saudi Arabia, where climate conditions differ and floods are more frequent. The flash flood occurred on the 8th of May 2014, and fortunately did not lead to extreme damage and caused only a single casualty [101]. Students at the university reported that the street water level was locally around 10cm, and the incident started at around 11PM local time.

This incident has been captured by the two sensors "8F48" and "D3CB". The water level estimates (using the days before May 6th as training data for both sensors) are illustrated in Figure 4.11 for both sensors. As can be seen from this Figure, the flood is clearly detected, and corresponds to a significant rise in the water level estimate. We can also see the onset of the flood is identical for both sensors, and that the estimated water levels are of the same order of magnitude (7 to 9 cm).

Unfortunately, the sink node has failed shortly after the flood occurred, around 11:30 PM. The failure of the sink node was probably caused by an electrical failure related to the flooding event. Nevertheless, the available data clearly shows the detection of a minor event shortly before midnight, 9th of May 2014, on both sensors. There was no way to detect the flood incident from the raw data of the ultrasonic, as illustrated in Figure 4.12 for one of the sensor nodes that encountered the flood (i.e. "8F48"). As shown by this Figure, naive thresholding on the raw water level measurement data would not allow one to detect the flood.

4.4 Implementation of ANN algorithms on microcontrollers

In order to validate our design and be able in practice to use our solution, we implement our algorithms on the dedicated microcontroller board and validate the performance as explained in the following sections.

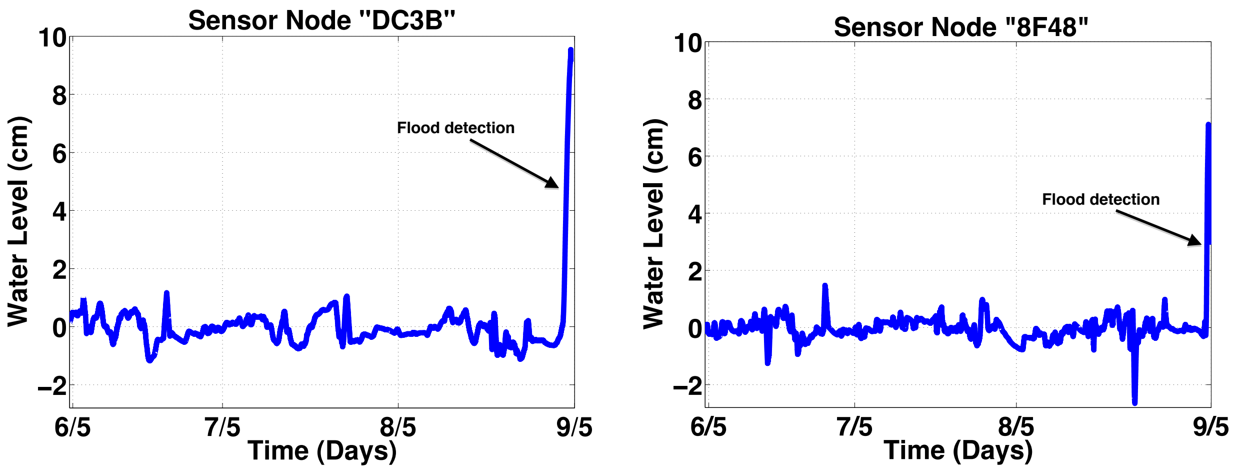


Figure 4.11: Water level estimation between May 6 and May 9, 2014, for sensor nodes ”DC3B” and ”8F48” deployed in Umm Al Qura University campus.

4.4.1 Dedicated sensing platform for flash flood monitoring applications

In the present case, we have built a customized hardware platform, described in the companion work [84]. This platform is built around a 32-bit microcontroller and illustrated in Figure 4.13. We selected for this application the `STM32F407`, a 32-bit `ARM Cortex-M4` based microcontroller from `ST Microelectronics` since it satisfies the requirements of our estimation problem and best trades off RAM, power consumption, cost and most importantly for this study, computation (since this sensor is sensing traffic flow, relaying and forwarding other messages in the sensor network, and estimating traffic flow conditions [83]).

4.4.2 On-Board Neural Network Algorithm

In order to build a neural network on the custom-based microcontrollers, we need to build not only the architecture but also the data storage system, for performing both training and prediction. The key values to be stored are weights associated with neurons, the weight updates and error gradients during the training phase.

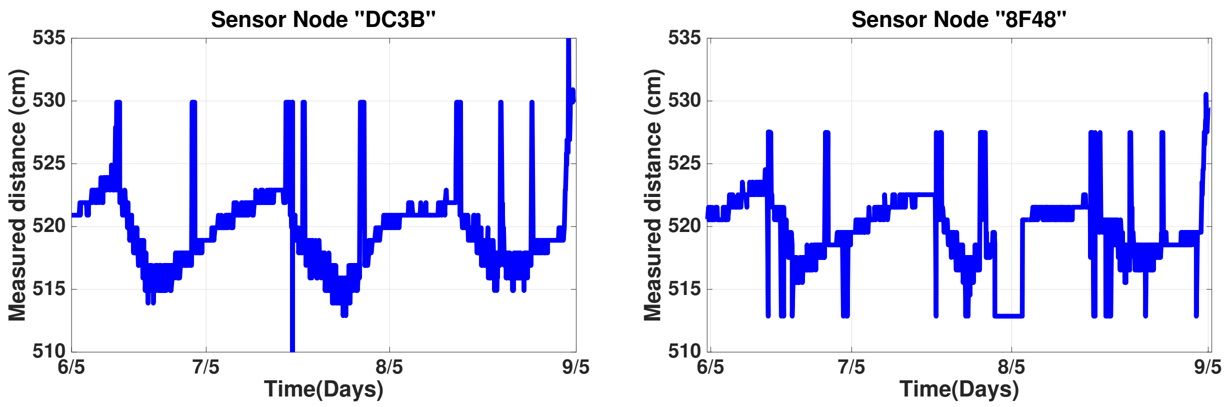


Figure 4.12: Raw ultrasonic measurements between May 6 and May 9, 2014, for sensor nodes "DC3B" and "8F48" deployed in Umm Al Qura University campus.

Initialization of the Neural Network:

To minimize the number of iterations in neural network training before convergence, the weights associated with neurons should be carefully initialized. Since our simulation results from training data set already show good estimation performance, and good spatio-temporal robustness, as shown in sections 4.3.5 and 4.3.6, the weights of the trained neural network of any given sensor is a good initial guess. Therefore, we initialize the neural network on board by the weights of the neural network trained in the simulation from training data between December 4th till December 21st.

Neural Network Training:

As we have illustrated previously in section 4.3.5, and in Figure 4.14, the trained NN model is robust when it is applied to long term prediction over two months until the error propagates due to the weather change. In order to continuously make accurate prediction, the NN model should be updated with the recently received data, which are saved in a 2GB micro SD card. The card has the capacity to host at most 12 months of data, which are sufficient for NN updating. Due to the limited computing power on board, updating the NN cannot be implemented in the batch mode introduced in section 4.2.3. The main reason is the repeatedly heavy computation of y_k in Equation 4.4 for all x in training data during the error propagation process. Thus

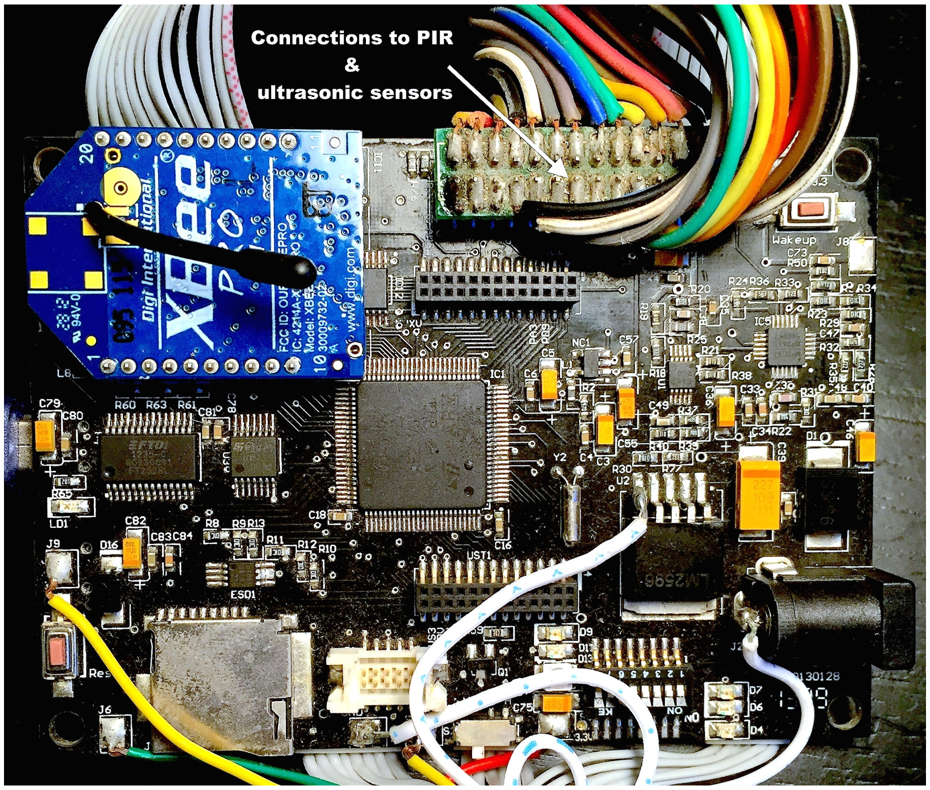


Figure 4.13: Custom-developed 32-bit microcontroller platform (9cm x 6.5cm) connected to an `XBee` module and to PIR and ultrasonic sensors.

we adopt the online training mode for updating the weights in NN model, due to its comparable performance to the batch mode but better efficiency. Instead of taking all x in training data for a single update of weights in NN, the online mode updates the weights by using x one by one. That is to say, the data saved in SD card flow into the NN individually and update it until the prediction error is reduced down below an acceptable threshold (e.g., 2 cm in the L_∞ case).

Implementation:

The implementation of code is done using `Keil v4.7` [102] which is a software provided by the `ARM` group, and optimized for C/C++ language. We have implemented our algorithms on the wireless sensor nodes using a conventional back-propagation neural network class in C language that makes use of gradient descent, with parameters defined as: 0.001 learning late, 1500 of maximum epochs during training, and

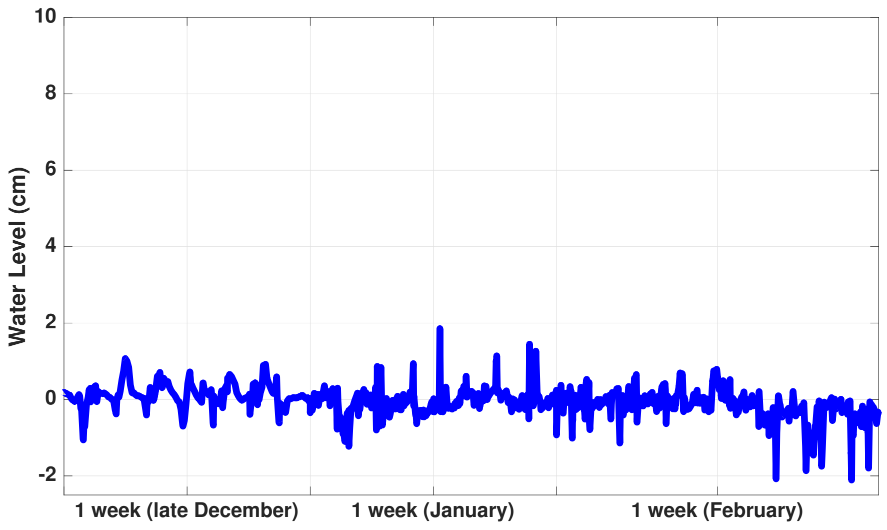


Figure 4.14: Prediction accuracy during different months on the sensor node ”A77D” installed on our campus in November 2013, using training data from December 2013 only. As can be seen from this Figure, the prediction performance slightly degrades in February, though its performance is still acceptable for flash flood monitoring applications.

maximum accuracy. The demonstration code is a 996 lines written in C++ language and is built on top of `<math.h>`, `<algorithm>`, `<fstream>`, `<string>`, `<vector>`, `<stdio.h>` libraries, as well as the designed neural network class `"neuralnetwork.h"`. The code can toggle between batch and online training mode, and gets the training stopping criteria from the user. Its total memory size (when compiled) is 101 kB (our microcontroller ROM size is 1 MB), while its peak memory usage is 58 kB (our microcontroller total RAM size is 192 kB), well below the limits of the `Cortex M4` microcontroller.

Neural Network performance:

We coded the neural network algorithm on the custom-based microcontroller platform described earlier, and tested them for real time performance. The prediction performance of the NN running on the computational platforms has similar results to the ones obtained on `Matlab`. However, since the platform has less computational capabilities than a computer, its computational time was expected to be much greater when we consider the training in addition to the prediction of the data in order to converge to the solution. We show the performance on the prediction of water level with retraining when the threshold is exceeded at the beginning of February 2014 in Figure 4.15. We used one week of data following the online training approach to retrain the NN parameters and as we can see this helped bringing the error to within 2 cm. The computational time measured for the process of training and predicting a week of data onboard with the online training approach took approximately, 118 minutes (11.2 minutes for prediction). We implement the on-board NN on the custom-based microcontroller platform, and evaluate its performance to see whether prediction error can be reduced after online updating. In Figure 4.14, we show the performance on the prediction of water level with training data from December 2013 only. It can be seen that the prediction performance slightly degrades in February 2014 (sometimes with errors greater than 2 cm). In Figure 4.15, we show the prediction with online training when the error exceeded 2 cm at the beginning of February 2014. We used one week of data following the online training mode to update the NN parameters. We can see that the error is reduced to be less than 2 cm.

Due to the limited computational capability of the low-power platform, the online training mode took about 2 hours for absorbing the one week data, while the prediction is quite fast 0.03 seconds per data sample. Therefore, the online training employed only one week data and stopped to proceed with more data to further reduce the error that is already less than 2 cm.

4.5 Discussion

The recent vast research activities in neural classification have established that neural networks are a promising alternative to various conventional classification methods. The advantage of neural networks lies in the following theoretical aspects. First,

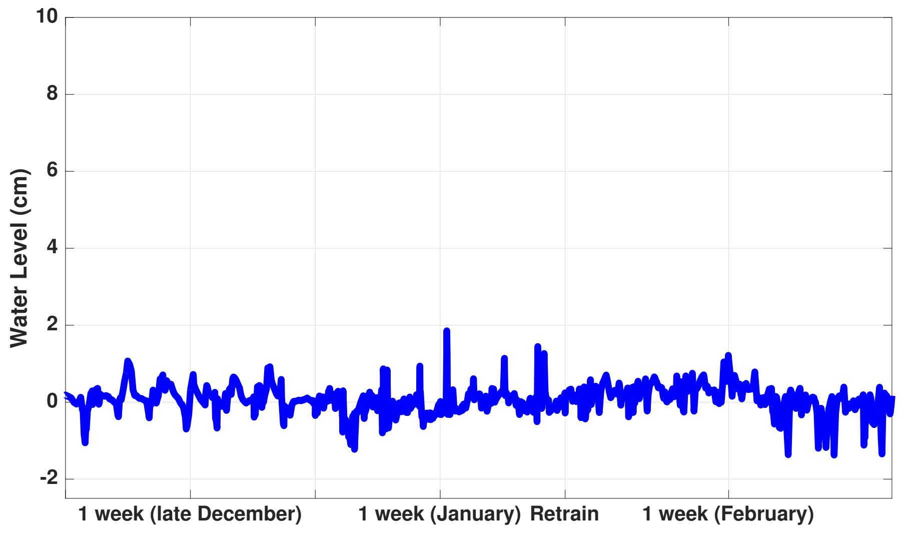


Figure 4.15: Prediction accuracy during different months on the sensor node "A77D" installed on our campus in November 2013, using training data from December 2013, with online on-board retraining of the parameters early February 2014. As can be seen from this Figure, the prediction performance has improved in February, compared to the results shown in Figure 4.14.

neural networks are data driven self-adaptive methods in that they can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model. Second, they are universal functional approximators in that neural networks can approximate any function with arbitrary accuracy. Third, neural networks are nonlinear models, which makes them flexible in modeling real world complex relationships. Finally, neural networks are able to estimate the posterior probabilities, which provides the basis for establishing classification rule and performing statistical analysis.

In the present case, ANNs exhibit good convergence properties, which enables us to use a low number of neurons and layers, making it suitable to a low power embedded systems application. In future work, we will investigate the possibility of using ANNs for detecting other information related to flooding, for example the presence of water on the ground, or the presence of rain, which may be inferred from air and ground temperature measurement patterns. Preliminary analysis on the flood data that we collected shows the possibility of using ANNs in this context, though more validation is needed.

4.6 Rain detection

In addition to monitoring water levels in streets, the proposed sensor could also detect rain events using temperature measurements. This information would be critical for estimating the propagation of floods in real time using hydrological models.

4.6.1 A Neural network classifier

We similarly use NNs to train a model for our classification problem (identifying whether rain events happen or not). The six-day data of ground and ambient temperature illustrated in Figure 4.16 include a rain event that happened on KAUST campus. Our target is to predict if it is raining or not at time t based on the ground and ambient temperatures at this time instant.

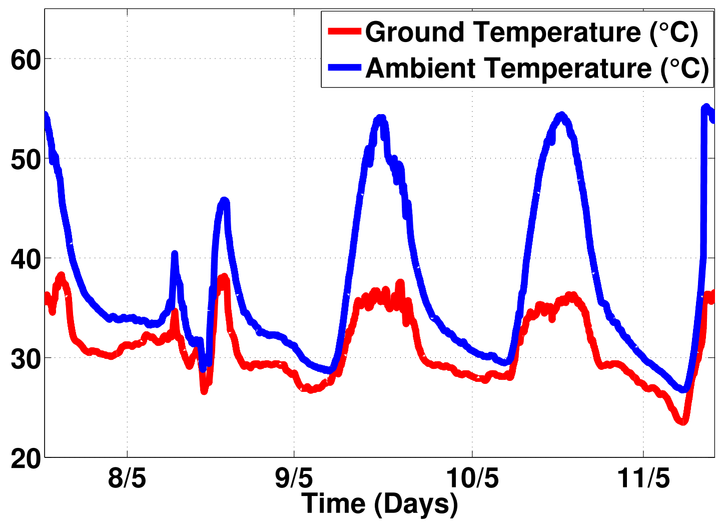


Figure 4.16: Ground and ambient temperature during May 7 to 11, 2014 in KAUST campus. The rain event on May 8th causes a strong drop in the measured ambient and ground temperatures.

As can be seen from Figure 4.16, a rain event can cause a strong drop in the ambient and ground temperatures. However, it is not feasible to directly report raining by monitoring the low values in temperatures, as temperatures follow a daily cycle. A NN with three inputs and ten neurons in one hidden layer is trained for a binary classification problem. The three inputs take the three features: ambient temperature, ground temperature and time. Binary class label indicates not raining (Class 1) and raining (Class 2).

We conducted the NN classification model on two different evaluations of the model to assess its robustness. The ground truth rain status is obtained from the local weather station in the area of Thuwal. We first divided the dataset randomly to training, validation and testing (70%, 15% and 15% respectively). For the second evaluation, we divided the dataset as follows: a training dataset containing 70% of rain events in a continuous period and 70% of no rain events in other continuous periods, as well as validation and testing datasets containing 15% of both rain and no rain events in different continuous periods. That is to say, the time span of training, validation and testing do not overlap. The purpose of the second evaluation is to check the time dependency of our model in identifying rain events since we have the time stamp for every sample as an input to our model (which affects the model creation since the recorded rain event happened at a particular time of the day).

4.6.2 Rain detection results

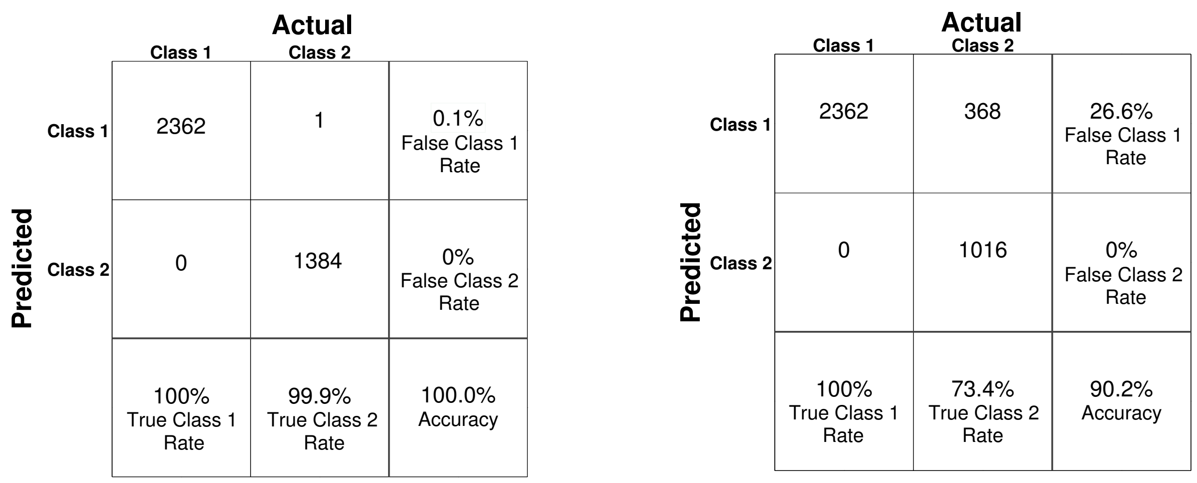
The performance of a NN classifier can be summarized in the confusion matrix concept. For the first evaluation (*i.e.,* with randomly divided training, validation and testing sets), the confusion matrix (shown in Figure 4.17 Left) indicates a high accuracy classification (very close to 100%) (accuracy in this case reflects how much "rain" and "non-rain" are correctly predicted). Only one rain sample is misclassified as "non-rain" leading to a recall (R) or true Class 2 (rain) rate (a.k.a. True Positive

Rate) of 99.9%. The precision (P), measuring how much predicted rain samples are truly rain samples, in this case is 1, since the 1384 samples predicted as Class 2 truly belong to Class 2 (no "non-rain" samples were classified as "rain" samples). We also evaluate our model by $F_{measure}$, which is a widely used metric for binary classification model. It considers both the precision and recall by taking a weighted average of them:

$$F_{measure} = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R} \tag{4.8}$$

where $\beta \geq 0$ is used to control the weight assigned to precision and recall. Usually, β is set to be 1 so that $F_{measure}$ is the harmonic mean of precision and recall. $F_{measure}$ close to 1 indicates the perfect predictive power of a classification model. We re-run the evaluation by different training, validation and testing sets randomly sampled from the whole data set, and achieve the similar confusion matrix, showing the stability of our model. Averaging on five independent runs, the $F_{measure}$ (or $F_1 - Score$ for $\beta = 1$) we got is 0.999.

For the second evaluation (*i.e.,* training, validation and testing subsets are cut from different time intervals), the classification performance remains good (illustrated in Figure 4.17 Right), though it slightly degrades to 90.2% of accuracy (on average on 5 runs). While the precision (P) is still 1, the recall (R) or the true Class 2 rate is degraded to 73.4% (on average on 5 runs). This means that we can trust the classification judgements made by our model, as all predicted rain events are truly belonging to the class of rain ($P = 1$). However, there are 26.6% rain events that remains unidentified. Given the single short raining period (8 hours) in our datasets, the current performance of rain detection is more than acceptable (with $F_1 - Score = 0.85$). The predictive power of the classification model would be improved when more rain samples under different conditions of temperature and time are available.



Predicted \ Actual	Class 1	Class 2	
Class 1	2362	1	0.1% False Class 1 Rate
Class 2	0	1384	0% False Class 2 Rate
	100% True Class 1 Rate	99.9% True Class 2 Rate	100.0% Accuracy

Predicted \ Actual	Class 1	Class 2	
Class 1	2362	368	26.6% False Class 1 Rate
Class 2	0	1016	0% False Class 2 Rate
	100% True Class 1 Rate	73.4% True Class 2 Rate	90.2% Accuracy

Figure 4.17: Left: Testing confusion matrix resulted from evaluation 1 when the dataset samples are randomly divided into training, validation and testing datasets. The classification is almost perfect with accuracy close to 100%. Right: Testing confusion matrix resulted from evaluation 2 for the time dependency analysis. The average accuracy of 5 runs is as high as 90.2%.

4.7 Chapter Remarks

In this part of the thesis work, an ANN approach is presented to contribute to the literature of flash flood sensing using a custom-designed sensor comprising an ultrasonic rangefinder and multiple passive infrared temperature sensors. Because of the extremely low absolute distance measurement error required by the system (on the order of 0.2%), one needs to estimate the temperature profile of the air layer between the sensor and the ground, as this temperature affects the speed of sound. Since this profile is impossible to model accurately (because of unknown model parameters and unknown boundary conditions), a non model-based approach is chosen for estimating the correction due to deviations in temperature showing that ANNs capture the effects of the underlying model very accurately, and can be used to monitor water level in streets in real time, given the important preprocessing step that contributes substantially to the accuracy of the estimations, this approach has been validated for a long testing period and the robustness analysis shows its stability over time and space. We also demonstrate that such algorithms can run in real time on low-power microcontroller platforms, greatly reducing power usage and bandwidth requirements in a wireless sensor network context.

Future work beyond the thesis work will be focused on the detection of rain using the reflections of the secondary lobes of the ultrasonic rangefinder. I will also investigate the detection of water presence (not thickness) on the ground using air and ground temperature measurements. This water presence information can be used for fault detection purposes, to make sure that the change in ground distance is actually caused by water.

Chapter 5

Inertial Measurement Units-Based Probe Vehicles: Trajectory and Traffic Conditions Estimation

Traffic sensing is a critical component of traffic estimation systems that generate traffic maps, travel time estimates, optimal routes for vehicles, or optimal control policies for traffic control systems. Traffic sensors can be broadly categorized in two types (as mentioned earlier in this thesis): Eulerian (fixed) sensors, which measure traffic conditions at a fixed point in space, and Lagrangian (mobile) sensors, which measure traffic conditions along the path of a vehicle. The former include a wide variety of sensors, including radars [26], inductive loop detectors [103], traffic cameras (including infrared cameras) [104] or magnetometers [21]. In contrast, Lagrangian sensors consist in positioning systems, most often the Global Positioning System (GPS), though other satellite-based systems could also be used. Other positioning systems have been used in the past, including for instance cellphone towers [105] for triangulation or trilateration of vehicles, though satellite based positioning systems are considerably more accurate than cellphone tower-based positioning. The GPS system can provide location information with a high precision, on the order of meters, depending on the configuration of the environment and on the type of GPS receiver.

Companies such as INRIX [106] leverage this new type of sensing, which can take the form of GPS-equipped participatory vehicles (such as the Mobile Millennium system, see [107]), GPS-equipped fleet vehicles [106] or the location tracks of cellphone users, obtained using trilateration [108]. Among all user location methods, GPSs or other satellite-based positioning systems are the most accurate. However, these sensing methods are not without drawbacks. In particular, the cost and the power consumption of GPSs is relatively high. In addition, the accuracy of GPS in urban areas can be severely degraded because of the urban canyon effect, which may reduce the number of visible satellites [109] [110] [111] and cause multi-path effects causing a loss of positioning accuracy [112]. This in turn makes it very difficult to precisely reconstruct the trajectory of a vehicle in a city. For example, a vehicle stopped after an intersection may appear to stop before the intersection, or a vehicle stopped in traffic is indistinguishable from a vehicle stopped at a curb (which happens frequently in a city), which complicates the processing of the data. For this reason, a very high number of probe vehicles is required to reliably estimate traffic conditions in cities.

An entirely different approach is the use different type of positioning systems that are immune to these positioning effects, such as Inertial Measurement Units (IMUs). IMUs are based on a combination of accelerometers and gyrometers, and can be used to determine the accelerations and rotation rates of a vehicle. IMUs do not require any external infrastructure and do not actively radiate. They also require an extremely low power to operate, considerably less than GPS or cellphone-based systems. Owing to their much lower complexity than GPS systems, IMUs are less expensive than them [113]. They do not require an antenna for receiving GPS signals, and are not at risk of losing connectivity with positioning satellites, which frequently happens with GPS systems. They are also immune to environmental noise effects, in particular to the multi-path effect encountered in cities. Because of their very high accuracy (over short time windows), IMUs are extremely good at detecting and classifying the type of congestion encountered (traffic light, stop and go waves, slow and continuous traffic) [114]. The resulting system is significantly less expensive than an all-GPS solution (because of the lower cost of IMU chips over GPS chips) and immune to noise or to GPS spoofing, thereby making it more reliable [115]. In addition, such a system offers strong guarantees for the privacy of the participating users [83] when used in conjunction with a short-range wireless sensor network.

Unfortunately, IMUs do not generate absolute position measurement data, and are only capable of estimating a trajectory in conjunction with the knowledge of one of the positions along the path. They are also prone to accumulating integration errors, and unless an absolute positioning system is used in conjunction with them, their trajectory estimate is bound to diverge.

The process of estimating the position of a vehicle using an Inertial Measurement Unit (IMU) is called dead reckoning. It somehow requires the determination of the attitude (orientation) of the device with respect to the Earth, and the integration of the acceleration and rotation rate measurements over time. This integration allows the estimation of the trajectory of an object equipped with an IMU, provided that the initial or final locations, velocities and orientations are known. Such a system is commonly used in aviation, in particular in navigation systems of commercial airplanes, which use high accuracy inertial measurements in conjunction with fixed ground beacons to estimate the location of the airplane in real time, even in the absence of GPS signals [116]. However, aviation-grade IMUs are considerably more accurate and more expensive than commercial MEMs IMUs used for this project, which requires new strategies to estimate the trajectories in the present situation. While ground vehicles are constrained to evolve in the transportation network (which helps in estimating their trajectories), the integration errors are too large to meaningfully estimate the velocity and position of a vehicle after a few seconds have elapsed. In addition, while aviation-grade IMUs are precisely aligned with the axes of the airplane during manufacturing, the installation of a low-cost IMU device inside a car cannot be done with the same standards. Therefore, the orientation of the device inside the vehicle is in general unknown, which greatly increases the complexity of the problem.

For all these reasons, the objective of the present part of the thesis is to address the following challenges:

- Self-calibration of the IMU, through the development of an auto-calibration algorithm applicable to ground vehicles
- Estimation of vehicle orientation, to compute the coordinate acceleration in the Earth frame
- Estimation of vehicle trajectory, using the kinematics of ground vehicles, and the fact that vehicles often stop (intersections, traffic lights) in urban environments.
- Path reconstruction using the road network.

The rest of this work is organized as follows. Section 5.1 describes the system used in this Chapter, and presents an actual hardware implementation of this system. Section 5.2 focuses on the problem of auto-calibration, and shows that the orientation of the sensing device can be reliably learned by the system after a few minutes of driving, allowing a very easy installation (the device only needs to be rigidly attached to the vehicle). Section 5.3.1 deals with the problem of estimating the attitude of the vehicle, a required step to compute the actual acceleration of the vehicle from the IMU measurements. Section 5.3.2 focuses on the problem of trajectory estimation in a city. In this section, we do not assume that the topology of the transportation network is known (this information can used by the device receiving the IMU data to match the position of the vehicle on a map), and specifically focuses on the problem of divergence in velocity estimation. We show that the velocity of the vehicle can be accurately estimated through a constrained quadratic program, corresponding to a L_1 regularized least squares problem. All numerical algorithms are validated with experimental data obtained from an IMU-equipped vehicle. Finally, in section 5.4, we conclude this work with the implementation of the path reconstruction using the road network to be able to infer the actual path where we propose a map matching approach based on a Bayesian formulation.

5.1 System Components

5.1.1 Traffic sensing principle

In this Chapter, we assume that the IMU devices onboard vehicles communicate with a fixed sensor network infrastructure, and exchange information with it locally. This local information exchange is requires to estimate the final position of the vehicle, before this data is exchanged, since no other absolute positioning information is available. Ideally, the range of the wireless system used in conjunction with IMU-based probe vehicles should be in the order of tens of meters. This data can for example be exchanged over `Bluetooth` or `WiFi`, and the endpoint can be a `Bluetooth` or `WiFi` reader.

Similar to the approach described in [83], this wireless sensor network approach offer strong theoretical privacy guarantees for the probe users, unlike current probe-based systems that send data to a centralized database.

In the proposed system, traffic conditions are inferred locally using the nodes of the fixed wireless sensor network. In addition to solving the user privacy problem, this wireless sensor network approach increases the reliability of the system (over server-based approaches), and reduces the total cost of the system (since the cost of IMU chips and short range ad-hoc transceivers is much lower than the cost of GPS chips and cellphone data transceivers).

5.1.2 Vehicular system

We use a custom-developed GPS/IMU system (Figure 5.1) based on a `Arm Cortex M4` processor operating at 168 MHz. It contains a 9-DOF IMU (accelerometer, gyrometer and magnetometer) and a GPS, and is powered through a USB port that is rigidly attached to the vehicle (through a car charger or a vehicle USB port), albeit at a random orientation with respect to the coordinates of the vehicle. The device can

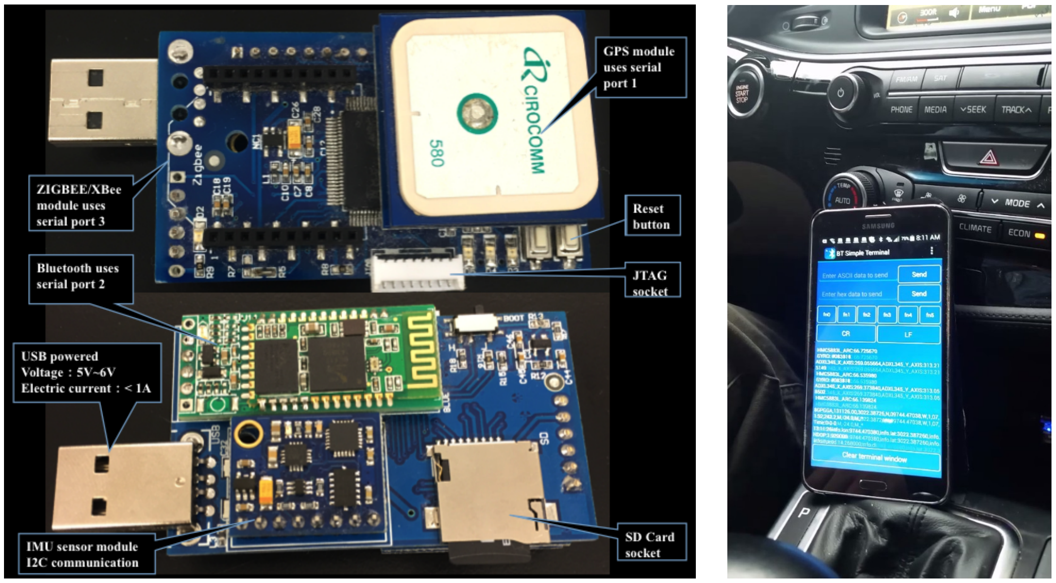


Figure 5.1: Left: custom-developed IMU board with bluetooth module. Right: real-time IMU data streaming from the IMU device to a Bluetooth enabled smartphone.

send data over a `IEEE 802.15.4 XBee` module transceiver or a `Bluetooth` transceiver (which is used in this study), at a 10Hz rate.

5.1.3 Fixed wireless sensor network system

The data generated by the IMU devices is sent to a wireless sensor network for processing. To minimize the latency of communications, the trajectory data generated by the IMUs is compressed before being sent to the wireless sensor network. The sensor network matches the estimated IMU trajectory to the actual road network, and sends the corresponding data to a traffic state estimation server. The complete system diagram [83] is illustrated in Figure 5.2.

5.1.4 Data processing

The proposed system is based in IMUs equipped vehicles (with GPS data used only for validation). Generating traffic measurement data from IMUs is nontrivial, and requires several processing steps at the vehicular sensor level and in the wireless sensor

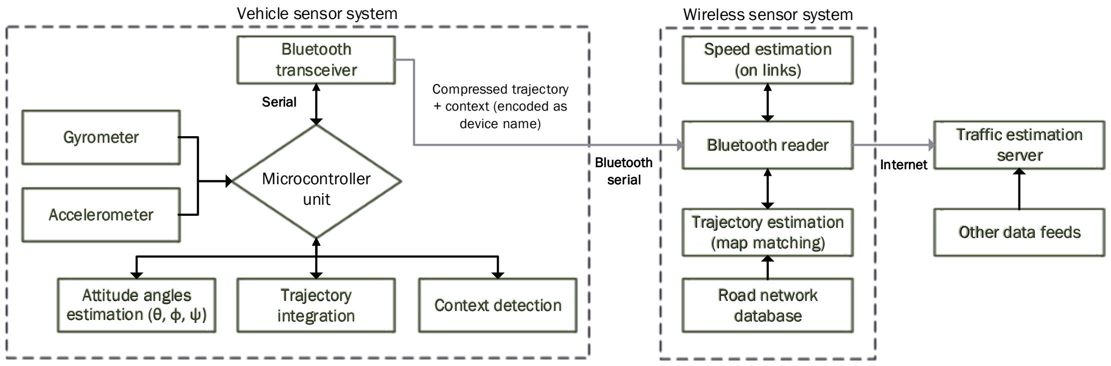


Figure 5.2: In the proposed system, traffic data from IMU is integrated to the fixed wireless sensor network, which computes the traffic maps using distributed computing. The resulting traffic maps are then forwarded to an output database

network. These processes are highlighted in Figure 5.2. The first step is to map the coordinates of the sensor to the coordinates of the vehicle, which will be referred to as automatic calibration in the remainder of the work. The resulting acceleration and rotation rate measurements from the sensor are mapped into the coordinates of the vehicle, and are used to determine the orientation of the vehicle with respect to the Earth. This allows us to compute the coordinate acceleration (in the Earth frame) by canceling the gravitational component of the acceleration. We then use the acceleration and rotation rate measurements to both estimate the yaw angle and the actual vehicle velocity. This combination allows us to generate vehicle trajectories, which are then compressed and sent to the wireless sensor network.

5.2 Automatic Calibration for attitude angles

Unlike GPSs, the orientation of an IMU sensor has an importance. To determine the trajectory of the vehicle, it is critical to determine the orientation of the IMU to measure the acceleration along the longitudinal, lateral and vertical axes of the vehicle. This could be achieved by carefully determining the orientation of the device in the vehicle (assumed to be constant, since the device is rigidly connected to an

USB port), and compute a corresponding rotation matrix mapping the coordinates of the device to the coordinates of the vehicle. However, this procedure is cumbersome, and prone to errors if the user removes and reinstall the sensor. To enable large scale deployments (with 200 IMU devices currently being manufactured for a pilot test in Texas), we have to determine the orientation of the device with respect to the vehicle automatically. Fortunately, the dynamics of ground vehicles is constrained, which allows us to develop an algorithm that automatically computes the rotation matrix transforming the vehicle coordinates into the vehicle coordinates.

5.2.1 Solution method

Since a rotation matrix is unitary, it is enough for us to determine how two axes are mapped through the rotation. Let i_c, j_c and k_c be the unit vectors associated with the longitudinal, lateral and vertical axes of the vehicle, as illustrated in Figure 5.7. Let us similarly define i_d, j_d and k_d as unit vectors associated with the longitudinal, lateral and vertical axes of the vehicle. Let $R_{d/c}$ be the rotation matrix mapping the coordinates of the vehicle into the coordinates of the device.

We assume that in average, the attitude of the vehicle on Earth is flat, that is the vehicle has on average a zero pitch and roll angle. This assumption is a standard assumption in aerospace, used to correct the long-term offsets of attitude indicators in airplanes, helicopters or drones, which are also assumed to fly in average with a zero pitch and roll attitude. Therefore, we have that

$$\frac{1}{T}\int_0^T \begin{bmatrix} a_x(t) \\ a_y(t) \\ a_z(t) \end{bmatrix} dt \approx \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{5.1}$$

where $a_x(t)$, $a_y(t)$ and $a_z(t)$ are the accelerations expressed in the vehicle coordinates, and g is the acceleration of gravity at the surface of the Earth.

Hence, if $\bar{a}_x(t)$, $\bar{a}_y(t)$ and $\bar{a}_z(t)$ represent the accelerations measured by the device, we have that

$$\frac{1}{gT}\int_0^T \begin{bmatrix} \bar{a}_x(t) \\ \bar{a}_y(t) \\ \bar{a}_z(t) \end{bmatrix} dt \approx R_{d/c} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.2}$$

In other words, the third column of $R_{d/c}$ is equal to $v_3 = \frac{1}{gT}\int_0^T \begin{bmatrix} \bar{a}_x(t) \\ \bar{a}_y(t) \\ \bar{a}_z(t) \end{bmatrix} dt$. The latter can be obtained by averaging the acceleration measurements of the IMU. In practice, the norm of v_3 may not be exactly equal to 1 (due to systematic errors of the accelerometer), and v_3 is renormalized to 1 as an extra step.

A single vector is not enough to determine the rotation matrix uniquely. Therefore, we need to determine the image of a second vector by $R_{c/d}$. This is achieved by determining the orientation of i_c in the sensor coordinates, as follows.

Let us first consider the projection of the acceleration vector on a plane perpendicular to v_3, that is, $\bar{a}_p(t) = \bar{a}(t) - < \bar{a}(t), v_3 > v_3$. The evolution of $\bar{a}_p(t)$ is illustrated in Figure 5.5.

As can be seen from this Figure, the residual acceleration is quite variable. Following classical kinematics, $R_{d/c}^{-1}\bar{a}_p(t)$ (corresponding to the coordinates of the projection of the acceleration in a plane perpendicular to the gravity vector) can be expressed as $\begin{bmatrix} \frac{dv(t)}{dt} \\ v(t)g_z(t) \end{bmatrix}$, where $v(t)$ is the velocity of the vehicle, and $g_z(t)$ corresponds to the rate of rotation of the vehicle around its vertical axis. This relationship assumes that the curvature radius of the trajectory $r(t)$ is related to $v(t)$ and $g_z(t)$ as $v(t) = r(t)g_z(t)$, which is only valid if the vehicle is not skidding, that is, if its velocity vector is constantly aligned with its longitudinal axis. Skidding is extremely rare in normal driving, and this assumption is not expected to be restrictive in practice.

While we do not have access to g_z (since we do not know the orientation of the device at this point), we can infer that g_z is almost equal to zero when the norm of the rotation rate vector (measured in the coordinates of the sensor) is very low. If we select the times for which $||g(t)||_2$ is under some threshold, the vector $\bar{a}_p(t)$ is almost equal to $\begin{bmatrix} \frac{dv(t)}{dt} \\ 0 \end{bmatrix}$, and thus, is aligned with the longitudinal axis of the vehicle. This is illustrated experimentally in Figure 5.5.

We thus have that $R_{d/c} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and $\bar{a}_p(t)$ are collinear. The only ambiguity left is the determination of the direction of the front of the vehicle. To achieve this, we use the integrated speed estimate, noting that the speed of a vehicle in forward mode is much higher than its speed in reverse. Hence, this allows us to determine the first column v_1 of $R_{d/c}$, by performing a linear fit on the values of $\bar{a}_p(t)$ such that $||g||_2$ can be neglected, and normalizing the corresponding vector (after identifying its direction using the speed integral criterion).

Having the first and the third column of the rotation matrix, the second column is obtained by cross product: $v_2 = v_3 \times v_1$, allowing us to determine the rotation matrix univocally. The complete process is outlined in Figure 5.3.

5.2.2 Implementation

We implemented the algorithm outlined above on the IMU boards described in section 5.1.2.

The setup is illustrated in Figure 5.4 for four different orientations.

Figure 5.6 illustrates the convergence of the acceleration with respect to the frame of the earth, computed after the optimization of the roll, pitch and yaw angles describing the matrix $R_{s/c}$ over time using a least square problem definition.

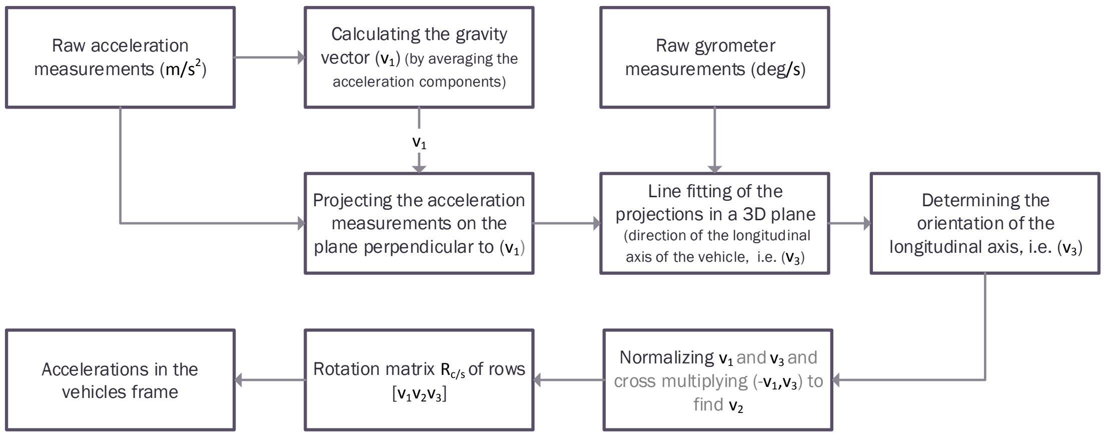

Raw acceleration measurements (m/s^2)
Calculating the gravity vector (v$_1$) (by averaging the acceleration components)
Raw gyrometer measurements (deg/s)
v$_1$
Projecting the acceleration measurements on the plane perpendicular to (v$_1$)
Line fitting of the projections in a 3D plane (direction of the longitudinal axis of the vehicle, i.e. (v$_3$)
Determining the orientation of the longitudinal axis, i.e. (v$_3$)
Accelerations in the vehicles frame
Rotation matrix R$_{c/s}$ of rows [v$_1$v$_2$v$_3$]
Normalizing v$_1$ and v$_3$ and cross multiplying (-v$_1$,v$_3$) to find v$_2$

Figure 5.3: The auto-calibration process.

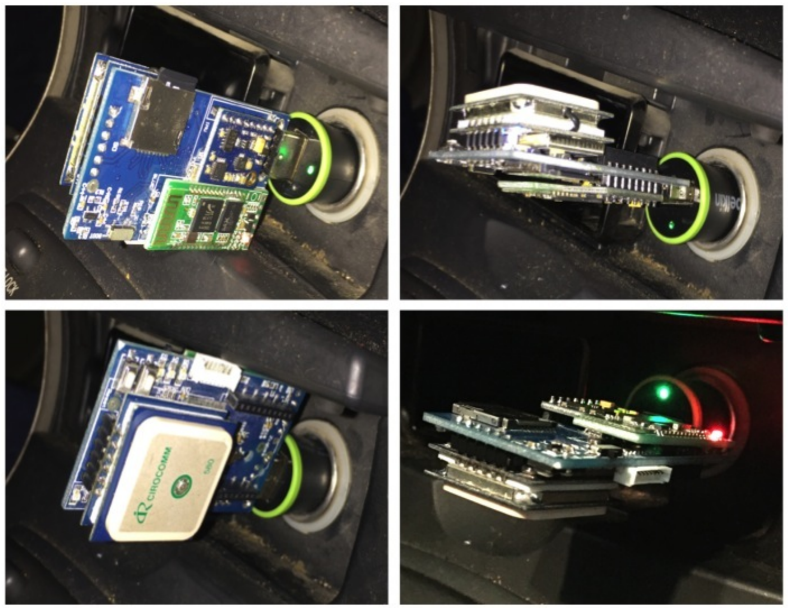


Figure 5.4: Testing for four different orientations using a custom designed IMU device plugged to USB port in a car.

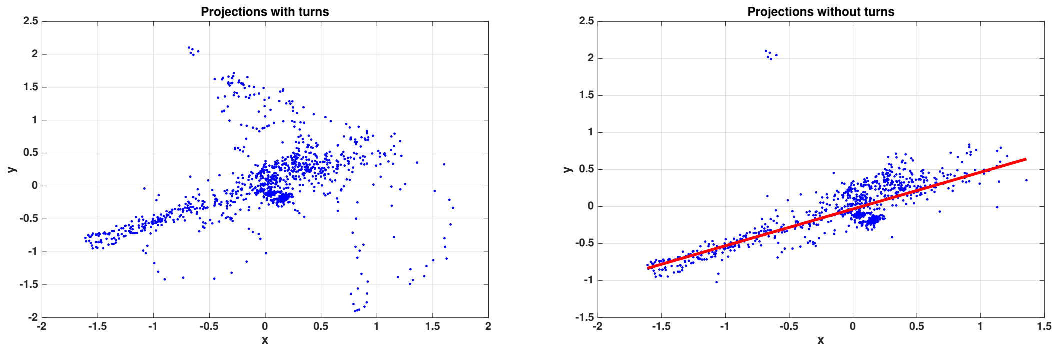


Figure 5.5: Projections of the accelerations measurements on the plane perpendicular to the gravity vector

5.3 Trajectory reconstruction

In this section, we focus on the problem of reconstructing the trajectory from inertial measurements only. We first focus on the computation of the acceleration of the vehicle in the earth frame, which somehow requires the computation of the attitude of the vehicle. The reason we need the attitude angles is that we need the coordinate acceleration, and not the proper acceleration (measured by the accelerometer), which corresponds to the acceleration measured with respect to a frame in free fall. The coordinate acceleration can be computed from the proper acceleration of the vehicle. We have that a_c=a_p-g, where g is the vector of acceleration due to gravity.

5.3.1 Attitude estimation algorithms

The direction cosine matrix (DCM) is generally used in attitude estimation and control of ground or air vehicles [117]. This filter is based on the following assumptions:

- The gyroscopes are used as the primary source of orientation information. The nonlinear differential kinematic equation that relates the time rate of change in the orientation of the vehicle to its rotation rate, and its present orientation, is integrated over time.

- The DCM filter uses reference vectors to detect the errors, and a proportional negative feedback controller between the sensed reference vectors (obtained from the accelerometer and the magnetometer) and the current estimate of the attitude. Precisely, the estimated rotation matrix R determining the attitude of the vehicle with respect to the Earth is computed as follows: $R(t+\Delta t) = \lambda R_{g,t,t+\Delta t} \times R(t) + (1-\lambda) R_{\mathrm{ref}}(t)$, where $R_{g,t,t+\Delta t}$ is the elementary rotation associated with the cumulated angular motion between t and $t+\Delta t$ and $R_{\mathrm{ref}}(t)$ is the rotation matrix obtained from the reference vectors. The coefficient λ trades off the current value of $R(t)$ with the current estimated reference rotation matrix $R_{\mathrm{ref}}(t)$, and is very close to 1 in practice.

- Based on the equation defined above, the matrix $R(t+\Delta t)$ will not be unitary (it is the linear combination of unitary matrices, which is itself not necessarily unitary). Therefore, small adjustments to the elements of the matrix $R(t+\Delta t)$ are required to preserve the unitarity of the matrix.

The matrix $R_{\mathrm{ref}}(t)$ can for instance be determined using the normalized acceleration vector $\frac{\mathbf{a}}{||a||_2}$ as a first reference vector, and the normalized projection of the magnetic field vector $\mathbf{b}$ on a plane perpendicular to a as a second reference vector. The elementary rotation matrix $R_{g,t,t+\Delta t}$ is equal to (assuming that the rotation is uniform between t and $t+\Delta t$:

$$R_{g,t,t+\Delta t} = \begin{bmatrix} 1 & -g_z\Delta t & g_y\Delta t \\ g_z\Delta t & 1 & -g_x\Delta t \\ -g_y\Delta t & g_x\Delta t & 1 \end{bmatrix} \tag{5.3}$$

Instead of expressing the attitude of the vehicle in terms of the rotation matrix $R(t)$, we can equivalently express it in terms of Euler angles: roll (ϕ), pitch (θ) and yaw (ψ). These angles are illustrated in Figure 5.7.

In the present situation, we run a DCM filter onboard the IMU device to determine the rotation matrix $R_{s/g}(t)$ transforming the ground coordinates into the sensor coordinates. Hence, we have that the gravity vector can be expressed in the sensor coordinates as $\mathbf{g}(t) = R_{s/g}(t)\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$. Therefore, the coordinate acceleration in the device frame becomes $a_p(t) - R_{s/g}(t)\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$, and the coordinate acceleration in the vehicle frame (which is required for the dead reckoning step becomes $R_{s/c}\times \left(a_p(t) - R_{s/g}(t)\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}\right)$, with $R_{s/c}$ the rotation matrix mapping the sensor coordinate to the vehicle coordinates determined by the automatic calibration step.

5.3.2 Dead Reckoning

Dead reckoning is defined as the process of determining the position of an object by projecting course and speed from a known past position and velocity [118]. In the present case, we are interested in estimating the trajectory of a vehicle using its final position, through measurements of accelerations and rotation rates on its path. The overall process used for dead reckoning (trajectory estimation) in the present case is illustrated in Figure 5.8.

Classically (for example in aerospace or in ship navigation), dead reckoning is performed by integration of the acceleration and rotation rate measurements, without any correction factor. This requires the operator to periodically perform position fixes (for example by using a ground-based beacon in aeronautics, or by measuring the attitude of a spacecraft using stars in astronautics). In the present situation, these position fixes are assumed to be unavailable. Therefore, we need to compensate the numerical errors induced by double integration using other methods.

Using classical kinematics results, the acceleration in a body frame moving horizontally has two components:

$a = (\frac{dv}{dt}, \frac{v^2}{r})$ where $\frac{dv}{dt}$ is the rate of variation of its velocity (the speed of the vehicle in the present case), and $\frac{v^2}{r}$ corresponds to a lateral acceleration component that depends on the radius of curvature r of its trajectory. In the present case, we are interested in mapping the trajectory of the vehicle in a 2D plane of axes X, Y.

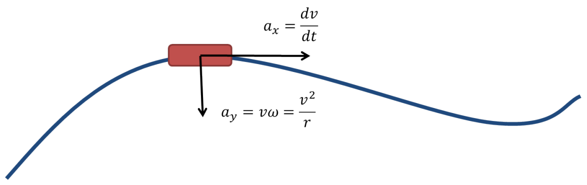


Figure 5.9: Accelerations in the frame of the vehicle, using classical kinematics. In this Figure, r represents the radius of the curvature of the trajectory.

Based on the above equation, the trajectory could theoretically be inferred directly from the longitudinal and lateral acceleration measurements. In practice, vehicles are turning at relatively low speeds, and the lateral acceleration measurement is not accurate enough to estimate the rotation rate $\omega_\psi = \frac{v}{r}$. We thus use the z-component of rotation rate vector (obtained from the gyroscope measurements), which is the derivative of the heading of the vehicle, and which corresponds to the rotation rate $\omega_\psi = \frac{v}{r}$ modulo some noise. The rotation rate is a much more reliable indicator of a vehicle turn.

Speed estimation:

The magnitude of the vehicle speed $\nu(t)$ is the integral of $\frac{dv(t)}{dt}$, that is, of the longitudinal component $a_x(t)$ of the acceleration. Therefore, we have $\nu(t) = \int_0^t a_x(\tau)d\tau$, which can be approximated in the present case as the Riemann sum

$$\nu(p \cdot \Delta t) = \sum_{k=0}^{p} a_x(k\Delta t)\Delta t \tag{5.4}$$

, assuming a zero velocity at the beginning of the experiment (stopped vehicle).

Without periodic velocity or position measurements, this numerical integration process will diverge due to sensor noise and integration errors. To address this, we need to somehow estimate the velocity or position of the vehicle at specific times, along its path. Since the vehicle is assumed to operate in a city, we can leverage the fact that most vehicles in cities stop at regular time intervals (for example at intersections, or traffic signals). In addition, the speed $v(t)$ during a turn can be estimated using the kinematic relation $a_y(t) = v(t)g_z(t)$, provided that the magnitude of $g_z(t)$ is sufficient. Therefore, unless the vehicle drives in a straight road and never stops at any point in time, we can periodically recalibrate the speed estimate over time.

To achieve this, we developed an optimization framework based on L_1 regularized least squares. Since the main contribution of the error of the accelerometer is accelerometer bias (which corresponds to a slowly time-varying signal), we define the decision variable as d, which corresponds to a set of coefficients $d_1, \ldots, d_n$ corresponding to piecewise constant approximations of the accelerometer bias. To simplify the problem, we assume that the time intervals used to define the piecewise constant drift are a multiple m of our integration step Δt. Therefore, at any time $k\Delta t$, the acceleration is given by $a_x(k\Delta t) - d_{\lfloor \frac{k}{m} \rfloor}$ With this assumption, equation (5.4) becomes:

$$v(p\Delta t) = \sum_{k=0}^{p} \left[a_x(k\Delta t) - d_{\lfloor \frac{k}{m} \rfloor} \right] \Delta t \tag{5.5}$$

which is linear in the decision variable d. The longitudinal acceleration measurements in the vehicle frame $a_x(\cdot)$ are not part of the optimization variable.

Once the speeds are expressed as linear function of the decision variable, we need to formulate the problem as a minimization of the discrepancy between observed speeds (when measurements are possible, that is, either during turns or during stops) and the speed obtained by integration. Let $\mathcal{M}$ correspond to the set of integer time instants for which velocity measurements are available. Minimizing the difference between estimated and measured velocities (when available) in the least squares sense amounts to solving the following unconstrained optimization problem:

$$\min_{d_1,\ldots,d_p} \sum_{i\in\mathcal{M}} (v(i\Delta t) - v_{\text{measured}}(i\Delta t))^2 + \lambda \sum_{j=1}^{p} |d_j - d_{j-1}| \tag{5.6}$$

The L_1 regularization term $\lambda \sum_{j=2}^{p} |d_j - d_{j-1}|$ is picked to enforce sparsity in the time variations of the drift vector. Indeed, for the MEMS accelerometers used in this study, drift is mainly caused by changes in temperatures, which would not happen very frequently. This leads us to an unconstrained L_1 regularized least squares problem that we can be solved efficiently through quadratic programming.

Let us now describe the obtention of velocity measurements along the path, at different time steps $i \in \mathcal{M}$.

Stop detection:

During stops, the rotation rates are almost zero (modulo the bias of the gyrometer) and the acceleration is almost constant on all axes. In addition, the longitudinal acceleration measured before a stop is negative (the car is decelerating). This allows us to distinguish the stop points from the constant speeds experienced along flat highways, for example. Based on these observations, we designed a simple algorithm to detect vehicle stops. The algorithm relies on the standard deviations of the last p acceleration components a_x, a_y and a_z. If the average of the standard deviations is less than some threshold, a vehicle stop is detected. This simple algorithm has been tested extensively on tens of minutes of data, and was able to detect each of the stops of the vehicle, without false stop detection. A subset of the corresponding test is shown in Figure 5.10, which illustrates that the rotation rate measurements generated by the gyro can similarly be used to detect vehicle stops.

Estimation of vehicle speed during turns:

The lateral acceleration measured during turns is related to the velocity of the vehicle during the turn, which allows us to obtain additional velocity measurements following the equation:

$$v_{\text{measured}}(t) = \frac{a_y(t)}{g_z(t)} \tag{5.7}$$

Since the term $g_z(t)$ (corresponding to the rotation rate around the vertical axis) appears in the denominator, the precision of the velocity estimate $v(t)$ depends on the magnitude of $g_z(t)$. If $g_z(t)$ is small (comparable to either the sensor quantization step or to the typical gyrometer bias), then the velocity estimate becomes inaccurate. Therefore, we only consider velocity measurements associated with large enough g_z in the above optimization problem. An example of speed estimate during turns can be seen in Figure 5.11. While the speed estimate during turns appears slightly noisy, it represents in average the correct vehicle velocity, as validated using the vehicle speedometer.

5.3.3 Validation

We conducted a series of tests to validate the L_1 regularized least squares optimization framework for velocity estimation. The results are illustrated in Figure 5.12 below.

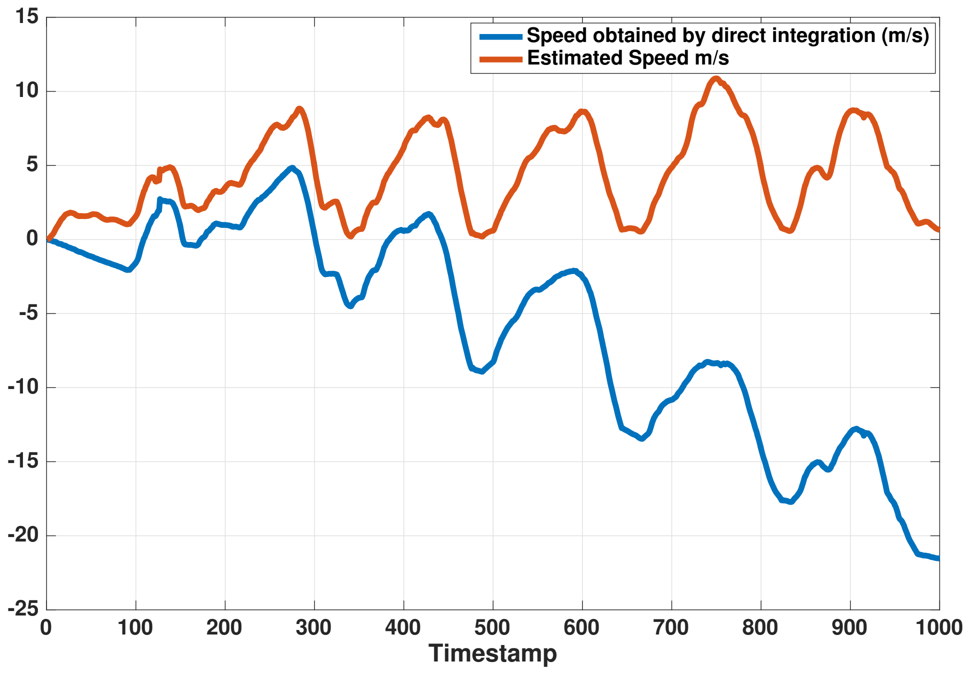


Figure 5.12: Blue: speed obtained by direct integration of the longitudinal acceleration. Red: estimated speed using the optimization framework

5.3.4 Yaw angle (heading) estimation

The estimation of the yaw angle $\psi(t)$ of the vehicle is much more straightforward than the estimation of the speed. Assuming that the pitch and roll angles of the vehicle remain relatively low, the yaw angle is calculated by integrating with respect to time the angular velocity measurements of the gyrometer with respect to the vertical axis of the vehicle:

$$\psi(t) = \int_0^t g_z(\tau)d\tau \tag{5.8}$$

The above equation can equivalently be rewritten as the Riemann sum $\psi(t) = \sum_{k=1}^{p} g_z(k\Delta t)\Delta t$.

Similarly to the velocity case, the yaw obtained by direct integration of the rotation rate measurement will diverge if not corrected with actual yaw measurements.

Unfortunately, the yaw cannot be estimated with the available sensors (accelerometer and gyrometer). While the actual yaw of the vehicle could be estimated with the magnetometer (detecting the magnetic North), the presence of metal in vehicles greatly affects the magnetometer readings, and obtaining an estimate do the heading is challenging, unless the magnetometer is calibrated for each vehicle (which is too cumbersome in practice).

However, the rotation rate can be filtered and sensor noise can be removed. Thus we had to investigate the signal we receive in order to best approximate the angles of turns. The rotation rate measurements are processed through different stages to filter the low and high noise sensor measurements. We first used a high-pass filter with a time constant of 30 seconds, then we apply a median filter to remove the noise. The results of processing the rotation rate signal are shown in Figure 5.13 where we represent the data of a trajectory of over 1200 samples. At the bottom of Figure 5.13, we show the result of our rotation rate integral to approximate our turning angle, i.e. the Yaw angle $\psi_{filtered}$, via the integral presented in equation 5.8.

5.3.5 Trajectory estimation and validation

The next step is to obtain the estimated path from the velocity and heading estimates. Since the vehicle is moving on a two dimensional plane (x,y), its position is given by:

$$x(t) = \nu(t)_{optimized} \cdot cos(\psi(t)_{filtered}) \cdot \Delta t + x(t-1) \tag{5.9}$$

$$y(t) = \nu(t)_{optimized} \cdot sin(\psi(t)_{filtered} \cdot \Delta t + y(t-1) \tag{5.10}$$

We have validated our trajectory estimation with several experiments in our university campus, as illustrated in Figure 5.14, we present some of them that included several turns, stops, uphills and downhills.

5.3.6 Piecewise linear trajectory approximation

Since the system is expected to operate in a dense urban environment, including a large number of probe vehicles, we want to compress the estimated vehicle trajectory before sending it to the fixed wireless network.

In order to keep the main features of the trajectory while minimizing the amount of data transmitted to the fixed sensor nodes, we use a piecewise linear approximation of the trajectory. The piecewise linear approximation aims to reconstruct the trajectory in the best possible way using only a limited number of linear components. It essentially amounts to a nonlinear constrained optimization problem, which can be stated as follows. Our objective is to find the values of the break points for n data points of (x,y) measurements (along the x and y axes) in which we get the new segments $f_{j,i_{fitted}}(x,y)$ (where j is from 1 to $m-1$; m is the number of end points of the segments and i is from 1 to n_j; n_j is the number of data points in the jth segment):

$$k_1 \leq f_{j,i_{fitted}}(x,y) < k_m$$

where k_1 and k_m are the beginning and the end points of the position vector on the x-axis(x) in which k_j represents the jth interior knot with coordinates (x_j, y_j). The objective function to be minimized is:

$$\min_{x_1,y_1,\ldots,x_{m-1},y_{m-1}} \quad \sum_{j=1}^{m-1}\sum_{i=1}^{n_j} |f_{j,i_{actual}}(x,y) - f_{j,i_{fitted}}(x,y)| \tag{5.11}$$

where $f_{j,i_{fitted}}(x,y)$ is the fitted piecewise linear function (segmentation) of $f_{j,i_{actual}}(x,y)$ (represents the actual position of the trajectory along the x and y-axes) given certain number of end points(m).

We choose a linear interpolation approach [119] between two break points, given as follows:

$$f_{j_{fitted}}(x,y) = (x, y_j + \frac{y_{j+1}-y_j}{x_{j+1}-x_j}(x-x_j)) \tag{5.12}$$

where $j = 1, \ldots, m-1$ is the segment index of $m-1$ segments required to fit the given trajectory(x_i, y_i). (x_j, y_j) are the x, y coordinates of the first break point (left end) of the j-th segment (i.e. k_j) and (x_{j+1}, y_{j+1}) are the x, y coordinates of the second break point (right end) of the j-th segment (i.e. k_{j+1}). Given the boundary conditions and the constraints of no pair of knots may lie too close to each other as well as the need to have the knots in increasing order, but still lie inside the first and final knot. This optimization problem can be solved for instance using the `fmincon` function of `Matlab` (which is part of the optimization toolbox).

This function finds the optimized interior knots by means of finding minimum of constrained nonlinear multivariable function (i.e. x and y position variables) starting at an initial estimate of having all the knots equally spaced. Figure 5.15 shows the result of the piecewise linearization of a path estimated by the dead reckoning algorithm. For this particular trajectory, we used 10 break points (in which we have 8 interior knots) to have 9 segments to optimize the distances between (x,y) measurements to (x,y) used in the fitting (i.e. $m = 10$ and $k_2, \ldots, k_9$).

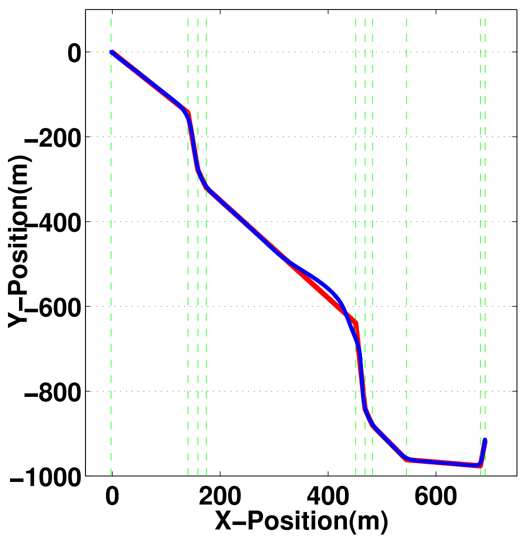


Figure 5.15: In red, the result of the piecewise linearization of the complete trajectory estimated by the dead reckoning algorithm.The blue curve shows the estimation of our dead reckoning algorithms and the green lines show the optimized segments using the piecewise linear optimizer.

5.4 Path reconstruction

5.4.1 Case Study

As can be seen from Figure 5.14, the estimated trajectory using the IMU data only is not compatible with the urban road network. While the estimated trajectory captures the main features of the actual trajectory, it exhibits increasing positioning error due to measurement uncertainty. In this section, we show that the knowledge of the road network structure can be used to reconstruct the actual vehicle path from the noisy trajectory data. As mentioned earlier, we consider a piecewise linear approximation of the trajectory, which consists of linear segments of given lengths. The lengths and heading changes between segments associated with the actual (linearized) trajectory is illustrated in Table 5.1. To reconstruct the actual trajectory, we consider the only data available to the fixed sensor beacon, *i.e.* the final location of the vehicle and the piecewise linear approximation of the trajectory (we consider only the last five segments for simplicity). We compare the last four segments of the piecewise linear trajectory approximation to all other possible paths that would lead to the measured final position for a certain trajectory. One possible way to do this is to construct a directed graph from local map data and compute all possible paths and routes to lead to this point from any point within a defined radius or area. We constructed this directed graph using the road network topology extracted from `Google Maps`. Based on this data, we found that 17 paths could lead to the destination point, and summarized the properties of these paths in Table 1. Since no absolute heading measurement data is available from our system, the link parameters are their length, and the heading change with respect to the previous link. We computed the corresponding parameters for the links corresponding to the piecewise linear approximation of the trajectory of the vehicle, and used these parameters to find the most likely path taken by the car. For this, we used a quadratic cost function defined by:

$$Cost = \sum_{i=1}^{4} (L_i - L_{DR_i})^2 + \sum_{i=1}^{4} (H_i - H_{DR_i})^2$$

where L_i corresponds to the length of the ith segment and H_i corresponds to the heading change of the segment i with respect to the segment $i-1$. L_{DR_i} and H_{DR_i} are the length and heading change parameters extracted from the trajectory reconstructed by the dead reckoning algorithm detailed above. As can be seen from Table 5.1, the path with the least cost is path 14, which corresponds to the actual path taken by the vehicle. , the concept may function by placing sensors at each area to receive those IMU data and construct the trajectory to identify previous routes taken to arrive to this point.

Path No.	Leg1		Leg2		Leg3		Leg4		Leg5		**Cost**
	Length	Heading	Length	Heading	Length	Heading	Length	Heading	Length	Heading	x(10^5)
1	98	N/A	110	-90	200	90	¿150	-15	-	-	1.93
2	300	N/A	10	-90	¿150	90	-	-	-	-	2.86
3	260	N/A	140	90	60	-90	10	-90	-	-	2.68
4	180	N/A	60	90	50	30	190	-30	150	-90	1.37
5	180	N/A	80	80	70	30	250	-30	65	-90	1.81
6	98	N/A	70	110	60	-30	70	20	45	-100	2.15
7	110	N/A	350	-110	130	-90	68	90	-	-	2.42
8	110	N/A	450	-110	94	90	48	-90	-	-	3.01
9	110	N/A	150	-110	98	90	150	-90	-	-	1.85
10	110	N/A	150	-110	200	90	130	-90	-	-	1.97
11	110	N/A	150	-110	300	90	89	-90	-	-	2.34
12	110	N/A	210	-20	39	-90	150	90	-	-	1.97
13	110	N/A	110	80	90	-20	190	-90	43	90	1.66
14	96	N/A	110	-90	100	-45	150	-30	100	30	0.93*
15	96	N/A	200	-90	250	-45	-	-	-	-	2.28
16	96	N/A	200	-90	100	-45	270	90	-	-	1.86
17	96	N/A	200	-90	100	-45	110	90	130	-90	1.17
DR	87	N/A	142	-96	102	-45	200	-38	400	40	-

Table 5.1: Sample of possible paths piecewise-linearized. The link parameters are their length (in meters) and their heading change (in degrees), where a positive heading change indicates a right turn and a negative heading change indicates a left turn.

5.4.2 Map matching based on a Bayesian formulation

Map matching algorithms identify the correct path on which a vehicle is traveling by integrating positioning data with the road network data. A map-matching algorithm could be used as an essential component to improve the performance of that support the navigation function of intelligent transport systems (ITS).

Many map matching algorithms have been developed by researchers [120] using different approaches such topological analysis of spatial road network data, probabilistic theory, Kalman filter, fuzzy logic, and belief theory.

However, this vast existing literature use the GPS data (with the characteristics: GPS localization accuracy and the sampling strategy), which give no advantage to infer the correct driving conditions, as well as the road conditions of the network map. Moreover, in our case, having the WSN parameters (*i.e.* received signal strength indicator (RSSI) and the node positions) add a lot of reliability to the overall system to work as a feedback system component of the ITS. Thus, we suggest the following novel map matching algorithm in order to find the most likely trajectory.

Every node should have the last three segments of every possible path to its location and since RSSI between the vehicle is considered between the vehicle and the node it makes it a useful parameter to consider in the following proposed Bayesian formulation [121] equation:

$$\phi(\tau_j/DR, RSSI) \propto \phi(\tau_j/RSSI) \times \phi(DR/\tau_j) \tag{5.13}$$

where DR is our dead reckoning estimation we explained in section 5.3.2, $RSSI$ is the received signal strength to a certain node (we consider the two closest nodes) from a vehicle and τ_j is a possible path a vehicle could take to a certain node in the network. So $\phi(\tau_j/DR, RSSI)$ is the probability distribution to maximize in order to find the best path and it is defined up to a scaling factor in order to integrate to 1.

$\phi(DR/\tau_j) \propto \phi(\tau_j/DR) \times \phi(DR)$ is a probability distribution function that aims to score every possible τ by comparing it to the dead reckoning estimation in an exhaustive search manner (this method is well know and could be made more efficient if we parametrize the paths) , and this is independent of RSSI, that is why one could cross the RSSI out of this term. The formula of the cost function that the probability distribution function is based on for a certain path:

$$Cost_{\tau_j} = \sum_{i=1}^{3} (L_{ji} - L_{DR_i})^2 + \sum_{i=1}^{3} (H_{ji} - H_{DR_i})^2$$

Thus, $\phi(DR/\tau_j) = 1 - \frac{Cost}{\sum_{j=1}^{N} Cost_j}$; N is the total number possible paths.

For $\phi(\tau_j/RSSI)$ we can use a machine learning approach that can give probabilities to the possible paths, learned from many trips (DRs) and takes into account the following parameters as features (inputs):

- The length of the path
- The speed limit of the path

- The number of stop signs along the path
- The number of lanes (in our case study two ways of a one path street)

After scoring all the possible routes, the most likely route shall give the maximum probability function *i.e.* τ^*.

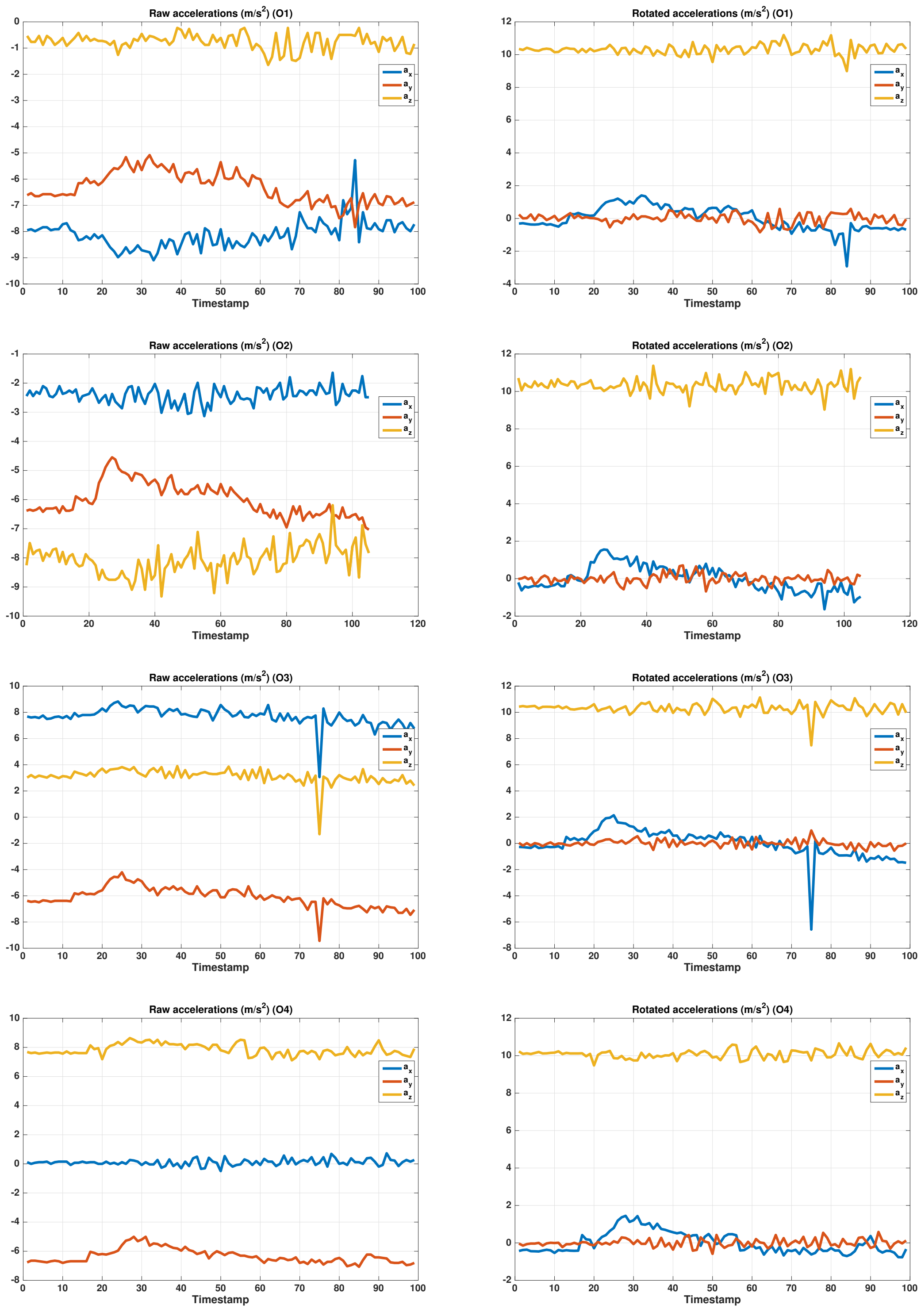


Figure 5.6: Accelerations with respect to the sensor frame (Left figures) in an orientation O_i and their corresponding rotated acceleration (Right figures) after the auto-calibration process for four different random orientation.

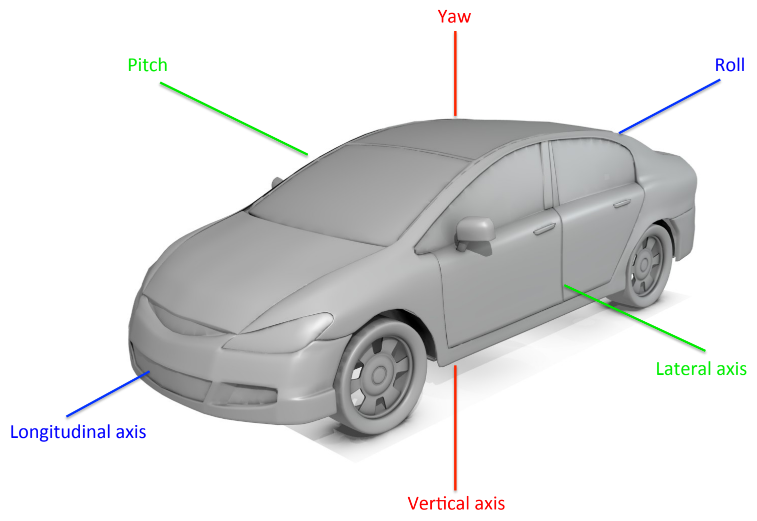


Figure 5.7: The attitude angles (roll (ϕ), pitch (θ) and yaw (ψ) and their corresponding axes

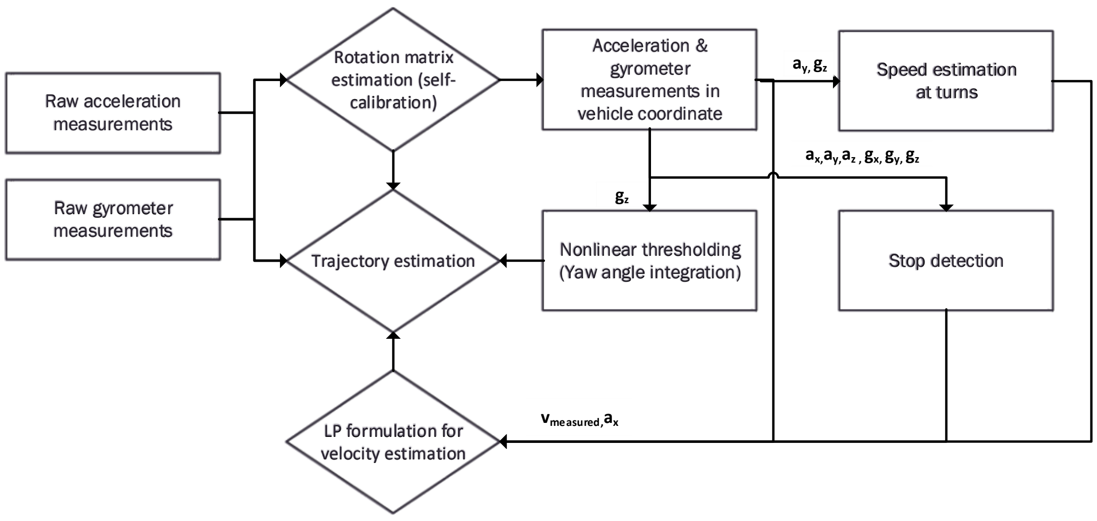

Raw acceleration measurements
Raw gyrometer measurements
Rotation matrix estimation (self-calibration)
Trajectory estimation
LP formulation for velocity estimation
Acceleration & gyrometer measurements in vehicle coordinate
Nonlinear thresholding (Yaw angle integration)
a_y, g_z
Speed estimation at turns
$a_x, a_y, a_z, g_x, g_y, g_z$
g_z
Stop detection
$v_{measured}, a_x$

Figure 5.8: Dead-reckoning principle. The input data from the IMU system in represented in the leftmost boxes.

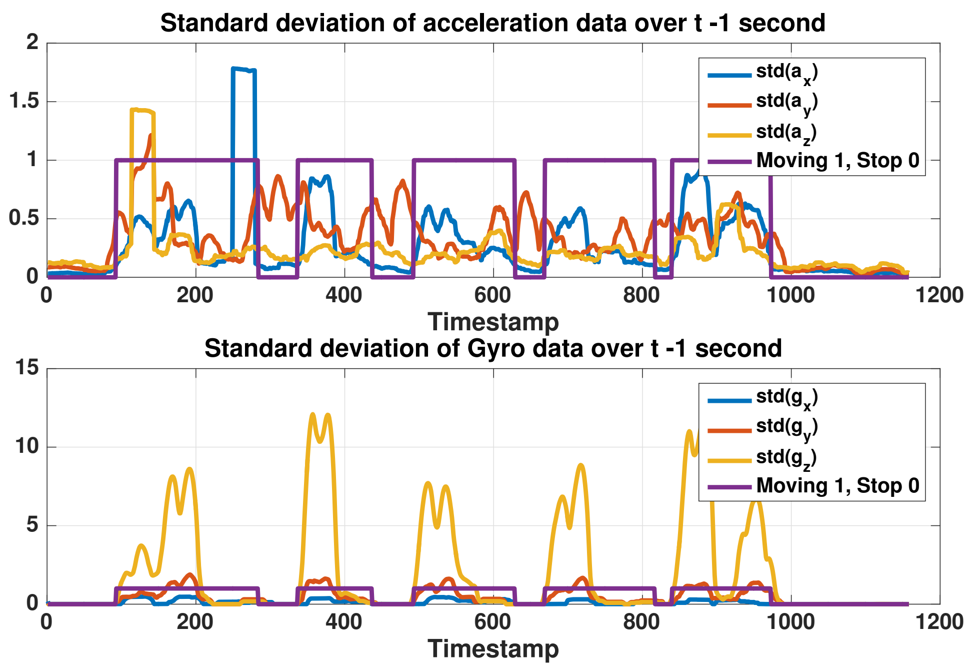


Figure 5.10: Stop detection measurement metrics represented in terms of standard deviation of acceleration and gyrometer data over a time window of 1s. The output of the thresholding filter is shown as a purple curve, and matches the actual vehicle stops.

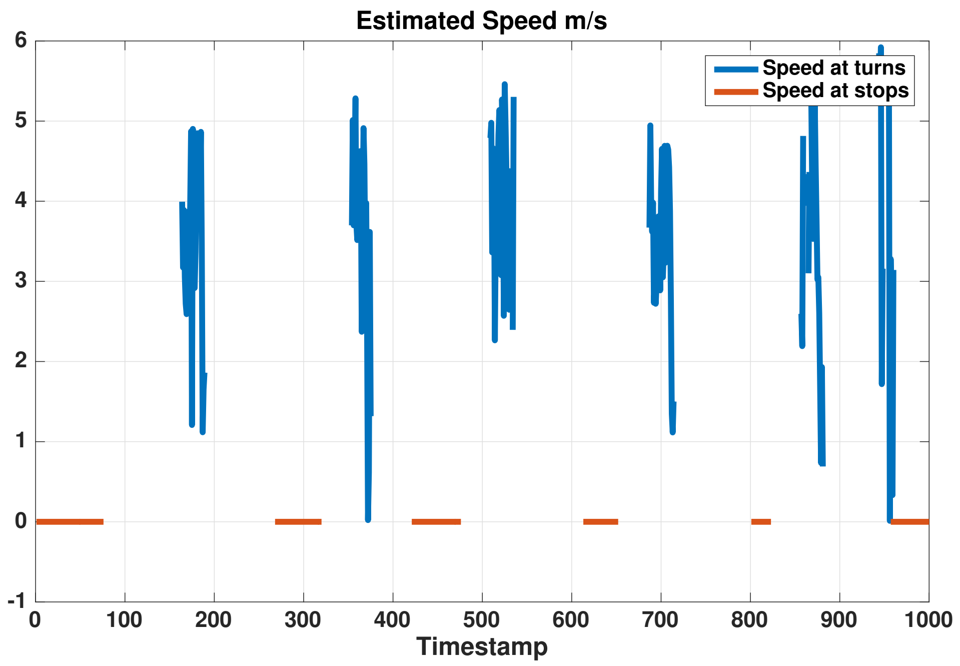


Figure 5.11: Estimated speeds at stops and turns, using the algorithms outlined above. It should be noted that the estimated speed during turns can be noisy, due to the increased vibrations of the vehicle during typical turns.

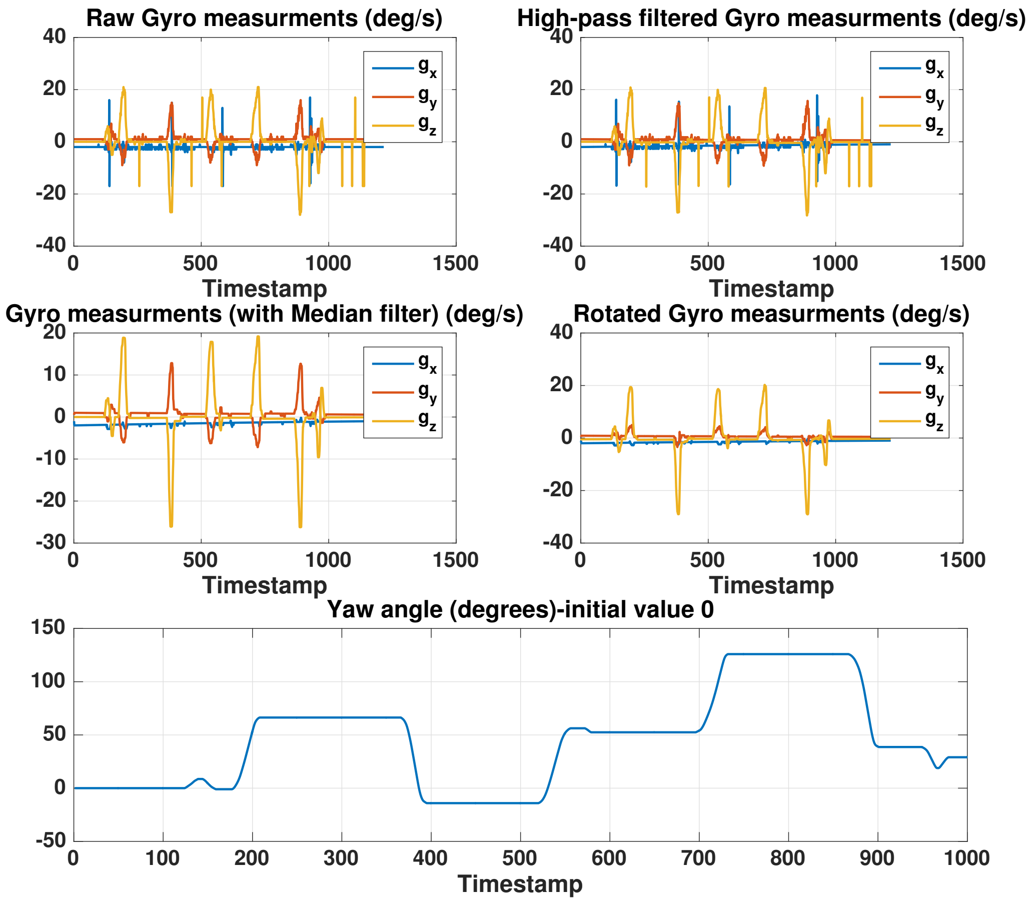


Figure 5.13: The process of filtering the rotation rate measurement for a certain experiment conducted in our university's campus. (Bottom) the yaw angle integral at initial point 0 degrees.

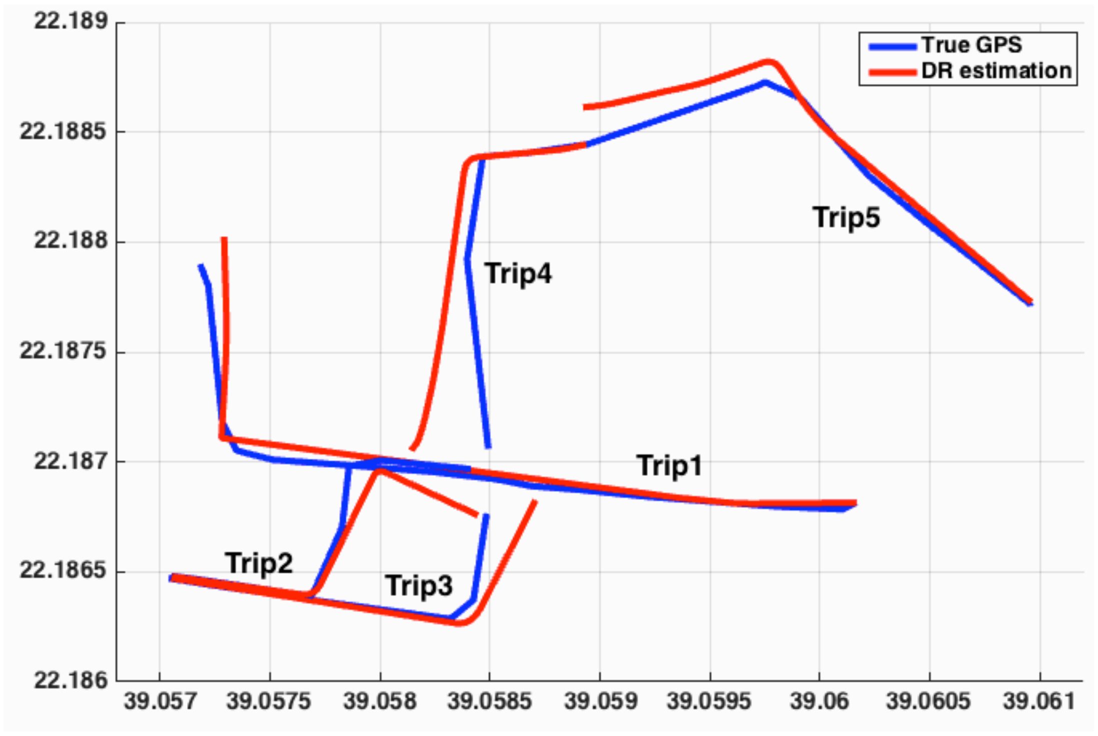


Figure 5.14: Trajectory estimation (dead reckoning) for some experiments/trips conducted with our custom based IMU device and trajectory estimation algorithms.

Chapter 6

Concluding Remarks

6.1 Summary

This thesis sums up the lessons learned from four years of WSN experiments and experience for smart city applications focusing on flash flood monitoring and traffic flow monitoring. In early experiments, the difficulties associated with operating a solar-powered WSN became evident, with hardware and software reliability issues and problems with the management of battery energy and solar power.

To address these early issues, we combined the lessons learned during our experiments to design a new hardware platform with greater computational and monitoring capabilities, which was subsequently used for the remainder of the thesis.

In Chapter 3, I described a set of tools for estimating the energy status of solar-powered wireless sensor networks nodes. Given the variability of solar energy availability and of the available battery capacity in a typical sensor network, we anticipate that such tools will be very useful to optimize sensing and network operations and to minimize the number of nodes that run out of energy. Even if ultra-low power wireless sensor networks do not necessarily need such an optimization (provided that their solar panels and batteries are oversized) to run, this suite of tools is very important for fault detection, isolation and most importantly to achieve energy-aware sensing computation and communication.

The high performance capabilities of the newly developed mote allow sophisticated estimation schemes to be implemented onboard for various sensing applications. We presented an ANN approach to flash flood sensing using a custom-designed flash flood sensor comprising an ultrasonic rangefinder and multiple passive infrared temperature sensors. Because of the extremely low absolute distance measurement error required by the system (on the order of 0.2%), one needs to estimate the temperature profile of the air layer between the sensor and the ground. Given the limited computational capabilities of the hardware platform, this profile cannot be modeled as a PDE. Therefore, we choose a non model-based approach for estimating the correction due to deviations in temperature. Our work shows that ANNs capture the effects of the underlying model very accurately, and can be used to monitor water levels in real time. We also showed that the same approach could be used to detect rain events.

Since a standalone flash flood sensor network is difficult to market, we chose to develop a dual flash flood/traffic sensing system, which can leverage the high performance of the newly developed platform. The dual ultrasonic-passive infrared sensor can also monitor traffic flow, though the density of sensors required to accurately sense traffic is much higher than the density of sensors required to accurately sense floods. Therefore, additional traffic sensing is needed, and since the fixed WSN architecture is already present (through the flash flood WSN), we chose to develop a system monitoring vehicles themselves (probe vehicles), and interacting with the fixed WSN.

Therefore I have developed a new system for probe sensing in urban environments that offers several advantages over current, satellite-based systems. Since it does not rely on satellite positioning data, it is immune to multi-path effects, which severely reduce the accuracy of positioning in cities. It is also less expensive (even when the cost of the fixed sensor nodes is factored in) than GPS-based systems, since IMUs are much more economical than GPS chips ($3/unit for IMUs vs. $25/unit for GPSs as of 2016). We also show that the accuracy of IMUs in conjunction with fixed wireless sensor systems used for periodic position updates is sufficient to perform traffic sensing operations. Another benefit of using IMUs instead of GPSs is that they work in very dense urban environments (with tunnels, underpasses and overpasses), and do not provide absolute location information, which greatly reduces the risk of privacy intrusion.

My Ph.D. work was focused on estimation of large-scale infrastructure system, which can leverage all forms of sensing, communication and computation including mobile sensing. This work can enable a suite of location-based services depending on flash flood and traffic flow measurement data. For example, whenever a flash flood occurs, routing emergency services, protecting the population or optimally responding to the flooding event require real-time flood and traffic data. This data can be further processed by servers to generate actual traffic flow and flash flood maps, by fusing the measurements to flood propagation or traffic propagation models. This thesis thus focuses on the sensing component of a dual traffic/flash flood information system.

6.2 Future Research Work

Future research work will focus on the development of such new scalable sensing systems, and develop sensor networks for congested urban areas to demonstrate their potential and cost effectiveness. There are a number of challenges to be addressed before sensing applications can be deployed at a large scale without causing any burden to the user and to the network.

One potential direction would be the development of a fully privacy preserving traffic sensor network, in which data is processed onboard nodes, as developed in the article [83]. Processing this data onboard computational nodes in a distributed fashion has never been done in the context of traffic sensing before, though recent advances in computational performance should allow this. The research I intend to carry is interdisciplinary by nature, and will involve embedded systems design applied to transportation and instrumentation systems as well as the development of novel computational methods for processing their measurement data.

Therefore, this research will be at the interface of computer sciences, electrical engineering and civil engineering. It will participate to the ongoing worldwide effort of developing smarter, more reliable and more resilient urban environment, known as Smart Cities.

REFERENCES

[1] E. A. Basha, S. Ravela, and D. Rus, "Model-based monitoring for early warning flood detection," in *Proceedings of the 6th ACM conference on Embedded network sensor systems.* ACM, 2008, pp. 295–308.

[2] R. Murty, G. Mainland, I. Rose, A. Chowdhury, A. Gosain, J. Bers, and M. Welsh, "Citysense: An urban-scale wireless sensor network and testbed," in *Technologies for Homeland Security, 2008 IEEE Conference on*, May, pp. 583–588.

[3] K. K. Khedo, R. Perseedoss, A. Mungur *et al.*, "A wireless sensor network air pollution monitoring system," *arXiv preprint arXiv:1005.1737*, 2010.

[4] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "The hitchhiker's guide to successful wireless sensor network deployments," in *Proceedings of the 6th ACM conference on Embedded network sensor systems.* ACM, 2008, pp. 43–56.

[5] `http://pems.eecs.berkeley.edu`.

[6] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse, "The hitchhiker's guide to successful residential sensing deployments," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems.* ACM, 2011, pp. 232–245.

[7] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental wireless sensor networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1903–1917, 2010.

[8] V. Dyo, S. A. Ellwood, D. W. Macdonald, A. Markham, C. Mascolo, B. Pásztor, S. Scellato, N. Trigoni, R. Wohlers, and K. Yousef, "Evolution and sustainability of a wildlife monitoring sensor network," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 127–140. [Online]. Available: http://doi.acm.org/10.1145/1869983.1869997

[9] F. Juraschek, A. Zubow, O. Hahm, M. Scheidgen, B. Blywis, R. Sombrutzki, M. Gunes, and J. Fischer, "Towards smart berlin-an experimental facility for

heterogeneous smart city infrastructures," in *Local Computer Networks Workshops (LCN Workshops), IEEE 37th Conference on.* IEEE, 2012, pp. 886–892.

[10] S. Coleri, S. Y. Cheung, and P. Varaiya, "Sensor networks for monitoring traffic," in *Allerton conference on communication, control and computing*, 2004, pp. 32–40.

[11] R. Lu, X. Lin, H. Zhu, and X. Shen, "Spark: a new vanet-based smart parking scheme for large parking lots," in *INFOCOM 2009, IEEE.* IEEE, 2009, pp. 1413–1421.

[12] R. Pantoni, C. Fonseca, and D. Brandão, "Street lighting system based on wireless sensor networks," 2012.

[13] A. Rowe, M. Berges, G. Bhatia, E. Goldman, R. Rajkumar, J. Garrett, J. M. F. Moura, and L. Soibelman, "Sensor andrew: Large-scale campus-wide sensing and actuation," *IBM Journal of Research and Development*, vol. 55, no. 1.2, pp. 6:1–6:14, 2011.

[14] M. K. Lindell and C. S. Prater, "Assessing community impacts of natural disasters," *Natural Hazards Review*, vol. 4, no. 4, pp. 176–185, 2003.

[15] ISDR disaster statistics, `http://www.unisdr.org/`.

[16] S. N. Jonkman and I. Kelman, "An analysis of the causes and circumstances of flood disaster deaths," *Disasters*, vol. 29, no. 1, pp. 75–97, 2005.

[17] M. Borga, E. Anagnostou, G. Blöschl, and J.-D. Creutin, "Flash flood forecasting, warning and risk management: the hydrate project," *Environmental Science & Policy*, vol. 14, no. 7, pp. 834–844, 2011.

[18] P. Milly, R. Wetherald, K. Dunne, and T. Delworth, "Increasing risk of great floods in a changing climate," *Nature*, vol. 415, no. 6871, pp. 514–517, 2002.

[19] M. Papageorgiou, "Applications of automatic control concepts to traffic flow modeling and control," 1983.

[20] Z. Jia, C. Chen, B. Coifman, and P. Varaiya, "The PEMS algorithms for accurate, real-time estimates of g-factors and speeds from single-loop detectors," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE.* IEEE, 2001, pp. 536–541.

[21] S.-Y. Cheung and P. P. Varaiya, *Traffic surveillance by wireless sensor networks: Final report*, 2007.

[22] C. M. Day, H. Premachandra, T. M. Brennan, J. R. Sturdevant, and D. M. Bullock, "Operational evaluation of wireless magnetometer vehicle detectors at signalized intersection," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2192, no. 1, pp. 11–23, 2010.

[23] T. N. Schoepflin and D. J. Dailey, "Algorithms for calibrating roadside traffic cameras and estimating mean vehicle speed," in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE.* IEEE, 2007, pp. 277–283.

[24] S. Erb, *Classification of vehicles based on acoustic features.* na, 2007.

[25] K. B. Eom, "Analysis of acoustic signatures from moving vehicles using time-varying autoregressive models," *Multidimensional Systems and Signal Processing*, vol. 10, no. 4, pp. 357–378, 1999.

[26] G. Alessandretti, A. Broggi, and P. Cerri, "Vehicle and guard rail detection using radar and vision data fusion," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 1, pp. 95–105, 2007.

[27] S. A. Ahmed, T. Hussain, and T. N. Saadawi, "Active and passive infrared sensors for vehicular traffic control," in *Vehicular Technology Conference, 1994 IEEE 44th.* IEEE, 1994, pp. 1393–1397.

[28] P. Zappi, E. Farella, and L. Benini, "Tracking motion direction and distance with pyroelectric IR sensors," *Sensors Journal, IEEE*, vol. 10, no. 9, pp. 1486–1494, 2010.

[29] Y. Jo and I. Jung, "Analysis of vehicle detection with WSN-based ultrasonic sensors," *Sensors*, vol. 14, no. 8, pp. 14 050–14 069, 2014.

[30] O. P. Tossavainen, J. Percelay, A. Tinka, Q. Wu, and A. Bayen, "Ensemble Kalman Filter based state estimation in 2d shallow water equations using Lagrangian sensing and state augmentation," in *47th IEEE Conference on Decision and Control, 2008.* IEEE, 2008, pp. 1783–1790.

[31] R. Alonso, M. Santillana, and C. Dawson, "On the diffusive wave approximation of the shallow water equations," *European Journal of Applied Mathematics*, vol. 19, no. 05, pp. 575–606, 2008.

[32] I. Sraj, K. T. Mandli, O. M. Knio, C. N. Dawson, and I. Hoteit, "Uncertainty quantification and inference of manning's friction coefficients using DART buoy data during the tōhoku tsunami," *Ocean Modelling*, vol. 83, pp. 82–97, 2014.

[33] J. V. Phillips and S. Tadayon, *Selection of Manning's roughness coefficient for natural and constructed vegetated and non-vegetated channels, and vegetation*

maintenance plan guidelines for vegetated channels in Central Arizona. Citeseer, 2006.

[34] L. Li, Y. Hong, J. Wang, R. Adler, F. Policelli, S. Habib, D. Irwin, T. Korme, and L. Okello, "Evaluation of the real-time TRMM-based multi-satellite precipitation analysis for an operational flood prediction system in Nzoia Basin, Lake Victoria, Africa," *Natural hazards*, vol. 50, no. 1, pp. 109–123, 2009.

[35] H. G. Sohn, J. Heo, H. Yoo, S. Kim, and H. Cho, "Hierarchical multi sensor approach for the assessment of flood related damages," *Proceedings of the 27th ISPRS congress*, 2008.

[36] Turn around, don't drown (US National Weather Service), `http://www.nws.noaa.gov /os/water/tadd/tadd-intro.shtml`.

[37] R. Fante, "Turbulence-induced distortion of synthetic aperture radar images," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 32, no. 4, pp. 958 –961, jul 1994.

[38] J. Krumm, "Inference attacks on location tracks," *Pervasive Computing*, pp. 127–143, 2007.

[39] L. Nachman, R. Kling, R. Adler, J. Huang, and V. Hummel, "The intel mote platform: A bluetooth*-based sensor network for industrial monitoring," vol. 2005, 2005, pp. 437–442, cited By (since 1996) 19. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-33645990555&partnerID=40&md5=2de14e5809595b88ffb7d322e90bbd49

[40] E. M. Shakshuki, H. Malik, and T. R. Sheltami, "Lessons learned: Simulation vs wsn deployment," in *Advanced Information Networking and Applications. AINA'09. International Conference on.* IEEE, 2009, pp. 580–587.

[41] V. Castelli, R. Harper, P. Heidelberger, S. Hunter, K. Trivedi, K. Vaidyanathan, and W. P. Zeggert, "Proactive management of software aging," *IBM Journal of Research and Development*, vol. 45, no. 2, pp. 311–332, March.

[42] L. I. R. Batteries, "Technical handbook."

[43] M. Ceriotti, M. Chini, A. Murphy, G. Picco, F. Cagnacci, and B. Tolhurst, "Motes in the jungle: lessons learned from a short-term WSN deployment in the Ecuador cloud forest," *Real-World Wireless Sensor Networks*, pp. 25–36, 2010.

[44] A. Ray, “Planning and analysis tool for large scale deployment of wireless sensor network,” *International Journal of Next-Generation Networks (IJNGN)*, vol. 1, no. 1, pp. 29–36, 2009.

[45] J. Luo and J.-P. Hubaux, “Joint mobility and routing for lifetime elongation in wireless sensor networks,” in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3. IEEE, 2005, pp. 1735–1746.

[46] A. Efrat, S. Har-Peled, and J. S. Mitchell, “Approximation algorithms for two optimal location problems in sensor networks,” in *Broadband Networks. BroadNets. 2nd International Conference on.* IEEE, 2005, pp. 714–723.

[47] F. Chen and R. Li, “Single sink node placement strategy in wireless sensor networks,” in *Electric Information and Control Engineering (ICEICE), 2011 International Conference on.* IEEE, 2011, pp. 1700–1703.

[48] J. Thelen, D. Goense, and K. Langendoen, “Radio wave propagation in potato fields,” in *1st Workshop on Wireless Network Measurements*, vol. 2. Citeseer, 2005, pp. 331–338.

[49] G. Anastasi, M. Conti, M. Di Francesco, and V. Neri, “Reliability and energy efficiency in multi-hop IEEE 802.15.4/ZigBee wireless sensor networks,” in *Computers and Communications (ISCC), IEEE Symposium on*, 2010, pp. 336–341.

[50] M. Mousa and C. Claudel, “Energy parameter estimation in solar powered wireless sensor networks,” in *Proceedings of the 2013 REALWSN conference.*

[51] T. Van Dam and K. Langendoen, “An adaptive energy-efficient mac protocol for wireless sensor networks,” in *Proceedings of the 1st international conference on Embedded networked sensor systems.* ACM, 2003, pp. 171–180.

[52] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, “Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks,” in *Proceedings of the 2nd international conference on Embedded networked sensor systems.* ACM, 2004, pp. 108–121.

[53] H. Huang, G. Hu, and F. Yu, “Energy-aware geographic routing in wireless sensor networks with anchor nodes,” *International Journal of Communication Systems*, vol. 26, no. 1, pp. 100–113, 2013.

[54] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, "Energy management in wireless sensor networks with energy-hungry sensors," *Instrumentation & Measurement Magazine, IEEE*, vol. 12, no. 2, pp. 16–23, 2009.

[55] L. M. Feeney, L. Andersson, A. Lindgren, S. Starborg, and A. Ahlberg Tidblad, "A testbed for measuring battery discharge behavior," in *Proceedings of the seventh ACM international workshop on Wireless network testbeds, experimental evaluation and characterization.* ACM, 2012, pp. 91–92.

[56] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, "Quanto: Tracking energy in networked embedded systems." in *OSDI*, vol. 8, 2008, pp. 323–338.

[57] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th workshop on Embedded networked sensors.* ACM, 2007, pp. 28–32.

[58] B. Saha and K. Goebel, "Modeling li-ion battery capacity depletion in a particle filtering framework," in *Proceedings of the annual conference of the prognostics and health management society*, 2009.

[59] C. Park, K. Lahiri, and A. Raghunathan, "Battery discharge characteristics of wireless sensor nodes: An experimental analysis," *power*, vol. 20, p. 21, 2005.

[60] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *Proceedings of the 4th international symposium on Information processing in sensor networks.* IEEE Press, 2005, p. 64.

[61] F. Bouabdallah, N. Bouabdallah, and R. Boutaba, "On balancing energy consumption in wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 6, pp. 2909–2924, 2009.

[62] G. Sikha, R. E. White, and B. N. Popov, "A mathematical model for a lithium-ion battery/electrochemical capacitor hybrid system," *Journal of The Electrochemical Society*, vol. 152, no. 8, pp. A1682–A1693, 2005.

[63] H.-L. Tsai, "Insolation-oriented model of photovoltaic module using matlab/simulink," *Solar Energy*, vol. 84, no. 7, pp. 1318 – 1326, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X1000160X

[64] M. Mousa and C. Claudel, "Energy parameter estimation in solar powered wireless sensor networks," in *Real-World Wireless Sensor Networks.* Springer, 2013, pp. 217–229.

[65] N. K. Suryadevara and S. C. Mukhopadhyay, "Wireless sensor network based home monitoring system for wellness determination of elderly," *Sensors Journal, IEEE*, vol. 12, no. 6, pp. 1965–1972, 2012.

[66] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment," in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks.* IEEE Computer Society, 2009, pp. 277–288.

[67] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca, "Shimmer–a wireless sensor platform for noninvasive biomedical research," *Sensors Journal, IEEE*, vol. 10, no. 9, pp. 1527–1534, 2010.

[68] A. H. Dehwah, M. Mousa, and C. G. Claudel, "Lessons learned on solar powered wireless sensor network deployments in urban, desert environments," *Ad Hoc Networks*, vol. 28, pp. 52–67, 2015.

[69] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture," in *Parallel and Distributed Processing Symposium. IPDPS, 20th International.* IEEE, 2006, pp. 8–pp.

[70] C. Alippi, R. Camplani, C. Galperti, and M. Roveri, "A robust, adaptive, solar-powered wsn framework for aquatic environmental monitoring," *Sensors Journal, IEEE*, vol. 11, no. 1, pp. 45–55, 2011.

[71] H.-C. Lin, Y.-C. Kan, and Y.-M. Hong, "The comprehensive gateway model for diverse environmental monitoring upon wireless sensor network," *Sensors Journal, IEEE*, vol. 11, no. 5, pp. 1293–1303, 2011.

[72] L. Hou and N. W. Bergmann, "Novel industrial wireless sensor networks for machine condition monitoring and fault diagnosis," *Instrumentation and Measurement, IEEE Transactions on*, vol. 61, no. 10, pp. 2787–2798, 2012.

[73] M. Castillo-Effer, D. H. Quintela, W. Moreno, R. Jordan, and W. Westhoff, "Wireless sensor networks for flash-flood alerting," in *Devices, Circuits and Systems, 2004. Proceedings of the Fifth IEEE International Caracas Conference on*, vol. 1. IEEE, 2004, pp. 142–146.

[74] E. Kuantama, L. Setyawan, and J. Darma, “Early flood alerts using short message service (sms),” in *System Engineering and Technology (ICSET), 2012 International Conference on.* IEEE, 2012, pp. 1–5.

[75] C. Lai, J. Yang, and Y. Chen, “A real time video processing based surveillance system for early fire and flood detection,” in *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE.* IEEE, 2007, pp. 1–6.

[76] H. Fu, X. Shu, A. Zhang, W. Liu, L. Zhang, S. He, and I. Bennion, “Implementation and characterization of liquid-level sensor based on a long-period fiber grating mach–zehnder interferometer,” *Sensors Journal, IEEE*, vol. 11, no. 11, pp. 2878–2882, 2011.

[77] N.-B. Chang and D.-H. Guo, “Urban flash flood monitoring, mapping and forecasting via a tailored sensor network system,” in *Networking, Sensing and Control, 2006. ICNSC’06. Proceedings of the 2006 IEEE International Conference on.* IEEE, 2006, pp. 757–761.

[78] M. Mousa, E. Oudat, and C. Claudel, “A novel dual traffic/flash flood monitoring system using passive infrared/ultrasonic sensors,” in *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on.* IEEE, 2015, pp. 388–397.

[79] M. Mousa and C. Claudel, “water level estimation in urban ultrasonic/passive infrared flash flood sensor networks using supervised learning,” in *Proceedings of the 13th international symposium on Information processing in sensor networks.* IEEE Press, 2014, pp. 277–278.

[80] V. Sakharov, S. Kuznetsov, B. Zaitsev, I. Kuznetsova, and S. Joshi, “Liquid Level Sensor Using Ultrasonic Lamb Waves,” *Ultrasonics*, vol. 41, no. 4, pp. 319–322, 2003.

[81] R. H. Brown, “Liquid Level Sensor,” Dec. 16 1997, uS Patent 5,697,248.

[82] C.-W. Lai, Y.-L. Lo, J.-P. Yur, and C.-H. Chuang, “Application of fiber bragg grating level sensor and fabry-perot pressure sensor to simultaneous measurement of liquid level and specific gravity,” *Sensors Journal, IEEE*, vol. 12, no. 4, pp. 827–831, 2012.

[83] E. Canepa, E. Odat, A. Dehwah, M. Mousa, J. Jiang, and C. Claudel, “A sensor network architecture for urban traffic state estimation with mixed eu-

lerian/lagrangian sensing based on distributed computing," in *Architecture of Computing Systems–ARCS 2014*. Springer, 2014, pp. 147–158.

[84] J. Jiang and C. Claudel, "A wireless computational platform for distributed computing based traffic monitoring involving mixed eulerian-lagrangian sensing," in *Industrial Embedded Systems (SIES), 2013-8th IEEE International Symposium on*. IEEE, 2013, pp. 232–239.

[85] D. Marioli, C. Narduzzi, C. Offelli, D. Petri, E. Sardini, and A. Taroni, "Digital time-of-flight measurement for ultrasonic sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 41, no. 1, pp. 93–97, 1992.

[86] J. C. Patra, P. K. Meher, and G. Chakraborty, "Development of laguerre neural-network-based intelligent sensors for wireless sensor networks," *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, no. 3, pp. 725–734, 2011.

[87] J. C. Patra, A. C. Kot, and G. Panda, "An intelligent pressure sensor using neural networks," *Instrumentation and Measurement, IEEE Transactions on*, vol. 49, no. 4, pp. 829–834, 2000.

[88] B. K. Bose, "Neural network applications in power electronics and motor drives an introduction and perspective," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 14–33, 2007.

[89] M. R. Meireles, P. E. Almeida, and M. G. Simões, "A comprehensive review for industrial applicability of artificial neural networks," *Industrial Electronics, IEEE Transactions on*, vol. 50, no. 3, pp. 585–601, 2003.

[90] F. Betin, A. Sivert, A. Yazidi, and G.-A. Capolino, "Determination of scaling factors for fuzzy logic control using the sliding-mode approach: Application to control of a dc machine drive," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 296–309, 2007.

[91] S. S. Ge and C. Wang, "Adaptive neural control of uncertain mimo nonlinear systems," *Neural Networks, IEEE Transactions on*, vol. 15, no. 3, pp. 674–692, 2004.

[92] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *Neural Networks, IEEE Transactions on*, vol. 7, no. 2, pp. 388–399, 1996.

[93] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[94] C. M. Bishop *et al.*, *Pattern recognition and machine learning.* springer New York, 2006, vol. 1.

[95] B. D. Ripley and R. M. Ripley, "Neural networks as statistical methods in survival analysis," *Clinical applications of artificial neural networks*, pp. 237–255, 2001.

[96] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis.* Springer, 1978, pp. 105–116.

[97] S. T. Chung, S. J. Kim, J. Lee, and J. M. Cioffi, "A game-theoretic approach to power allocation in frequency-selective gaussian interference channels," in *in Proc. IEEE International Symposium on Inform. Theory, Pacifico.* Citeseer, 2003.

[98] M. Schmidt, "Least squares optimization with l1-norm regularization," *CS542B Project Report*, 2005.

[99] B. Bhattacharya and D. P. Solomatine, "Neural networks and m5 model trees in modelling water level–discharge relationship," *Neurocomputing*, vol. 63, pp. 381–396, 2005.

[100] V. B. Veiga, Q. K. Hassan, and J. He, "Development of flow forecasting models in the bow river at calgary, alberta, canada," *Water*, vol. 7, no. 1, pp. 99–115, 2014.

[101] Saudi flash floods, `http://www.emirates247.com/news/region/saudi-flash-floods-one-killed-in-makkah-2014-05-10-1.548596`.

[102] ARM uvision Keil software, `http://www2.keil.com/mdk5/uvision//`.

[103] Y.-K. Ki and D.-K. Baik, "Vehicle-classification algorithm for single-loop detectors using neural networks," *Vehicular Technology, IEEE Transactions on*, vol. 55, no. 6, pp. 1704–1711, 2006.

[104] M. Bramberger, J. Brunner, B. Rinner, and H. Schwabach, "Real-time video analysis on an embedded smart camera for traffic surveillance," in *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE.* IEEE, 2004, pp. 174–181.

[105] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter *et al.*, "Place lab: Device positioning using radio beacons in the wild," in *International Conference on Pervasive Computing.* Springer, 2005, pp. 116–133.

[106] D. Schrank, T. Lomax, and S. Turner, "Tti's 2010 urban mobility report powered by inrix traffic data," *Texas Transportation Institute, The Texas A&M University System*, vol. 17, 2010.

[107] D. Work and A. Bayen, "Impacts of the mobile internet on transportation cyberphysical systems: traffic monitoring using smartphones," in *National Workshop for Research on High-Confidence Transportation Cyber-Physical Systems: Automotive, Aviation, & Rail*, 2008, pp. 18–20.

[108] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems.* ACM, 2008, pp. 323–336.

[109] L. Ojeda and J. Borenstein, "Personal dead-reckoning system for gps-denied environments," in *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, Sept 2007, pp. 1–6.

[110] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[111] A. Bachrach, S. Prentice, R. He, and N. Roy, "Range–robust autonomous navigation in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, 2011.

[112] W. Chen, Z. Li, M. Yu, and Y. Chen, "Effects of sensor errors on the performance of map matching," *Journal of Navigation*, vol. 58, no. 02, pp. 273–282, 2005.

[113] A. Jimenez, F. Seco, C. Prieto, and J. Guevara, "A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu," in *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, Aug 2009, pp. 37–42.

[114] S. Wan and E. Foxlin, "Improved pedestrian navigation based on drift-reduced mems imu chip," in *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation*, 2001, pp. 220–229.

[115] Y. Fuke and E. Krotkov, "Dead reckoning for a lunar rover on uneven terrain," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 411–416.

[116] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 425–437, 2002.

[117] W. Premerlani and P. Bizard, "Direction cosine matrix imu: Theory," *DIY Drones.[Online][Cited: 1 7 2012.] http://diydrones. ning. com/profiles/blogs/dcm-imu-theory-first-draft*, 2009.

[118] A. Jimenez, F. Seco, C. Prieto, and J. Guevara, "A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu," in *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on.* IEEE, 2009, pp. 37–42.

[119] G. Barequet and M. Sharir, "Piecewise-linear interpolation between polygonal slices," *Computer vision and image understanding*, vol. 63, no. 2, pp. 251–272, 1996.

[120] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 312–328, 2007.

[121] A. Gelman, "A bayesian formulation of exploratory data analysis and goodness-of-fit testing*," *International Statistical Review*, vol. 71, no. 2, pp. 369–382, 2003.

7 Accepted Papers

• Mustafa Mousa, Christian Claudel, "Energy parameter estimation in solar powered wireless sensor networks", *Published in ACM RealWSN conference (Lectures Notes in Electrical Engineering),Springer*, September 2013.

• Ahmad Dehwah, Mustafa Mousa, Edward Canepa, Enas Odat, Jiming Jiang and Christian Claudel, "Poster Abstract: Enhancing user privacy in probe-based traffic monitoring systems using distributed computing", *Presented in European Wireless Sensor Networks (EWSN) conference*, February, 2013.

• Ahmad Dehwah, Mustafa Mousa, Christopher Knox, Christian Claudel, "Poster Abstract: Sadeem: a solar-powered wireless sensor network testbed for energy modeling in urban, desert environments", *Presented in RTSS conference*, December, 2012.

• Mustafa Mousa, Ahmad Dehwah, Christian Claudel, , "Poster Abstract: Experimental analysis of environmental perturbations on wireless sensor network operation", *Presented in European Wireless Sensor Networks (EWSN) conference*, February, 2014.

• Edward Canepa, Enas Odat, Ahmad Dehwah, Mustafa Mousa, Jiming Jiang and Christian Claudel "A sensor network architecture for urban traffic state estimation with mixed Eulerian/Lagrangian sensing based on distributed computing", *Submitted to ARCS conference,Springer*, April, 2014.

• Mustafa Mousa, Christian Claudel, "Poster Abstract: Water Level Estimation in Urban Ultrasonic/Passive Infrared Flash Flood Sensor Networks Using Supervised Learning", *Published in Information processing for sensor networks (IPSN) conference, IEEE*, April, 2014.

• Ahmad Dehwah, Mustafa Mousa and Christian G. Claudel, "Lessons Learned on Solar Powered Wireless Sensor Network Deployments in Urban, Desert Environments", *Published in AD HOC Network Journal (Elsevier)*, April, 2014.

• Mustafa Mousa, Mohammed Abdulaal, Stephen Boyles, Christian Claudel, "Inertial Measurement Unit-based Traffic Monitoring Using Short Range Wireless Sensor Networks", *Accepted and presented in Transportation Research Board (TRB) conference*, Janurary, 2015.

• Mustafa Mousa, Mohammed Abdulaal, Stephen Boyles, Christian Claudel, "Wireless sensor network-based urban traffic monitoring using inertial reference data", *Accepted and presented in the Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on. IEEE*, June, 2015.

• Mustafa Mousa, Enas Oudat, Christian Claudel, "A Novel Dual Traffic/Flash Flood Monitoring System Using Passive Infrared/Ultrasonic Sensors", *Accepted and presented in the Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on. IEEE*, October, 2015.

• Enas Oudat, Mustafa Mousa, Christian Claudel, "Vehicle Detection and Classification Using Passive Infrared Sensing", *Accepted and presented in the Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on. IEEE*, October, 2015.

• Mustafa Mousa, Kapil Sharma, Christian Claudel, "Poster Abstract: Automatic Calibration of Device Attitude in Inertial Measurement Unit Based Traffic Probe Vehicles", *Accepted and presented in the Information processing for sensor networks (IPSN) conference, IEEE*, April, 2016.

• Mustafa Mousa, Xiangliang Zhang, Christian Claudel, "Flash Flood Detection in Urban Cities Using Ultrasonic and Infrared Sensors", *Accepted to IEEE sensors Journal*, June 2016.

7 Submitted/In submission Papers

• Mustafa Mousa, Kapil Sharma, Mohammed Abdulaal, Christian Claudel, “Inertial Measurement Units Based Probe Vehicles: Trajectory and Traffic Conditions Estimation”, *Submitted to Elsevier, Transportation Research Part C*, 2016.

• Mustafa Mousa, Kapil Sharma, Mohammed Abdulaal, Steve Boyles, Christian Claudel, “Inertial Measurement Units Based Probe Vehicles: Map Matching Approach for Path Reconstruction”, *In preparation to IEEE transaction on Intelligent Transportation Systems*, 2016.

7 Patents

- Mustafa Mousa, Mohammed Abdulaal, Christian Claudel, “Inertial measurement unit-based traffic flow monitoring”, *Pending*, 2015.
- Edward Canepa, Christian G Claudel, Atif Shamim, Ahmad H Dehwah, Mustafa Mousa, Jiming Jiang, “System and method for monitoring traffic while preserving personal privacy (CA 2881198 A1) ”, *Granted*, 2015.