More than downloading

Toward a scale with wireless connectivity

ABDEL AHMID

KTH Information and Communication Technology

Degree project in
Communication Systems
First level, 15.0 HEC
Stockholm, Sweden

More than downloading

Toward a scale with wireless connectivity

Abdel Ahmid

16 June 2012

Bachelor's thesis

Mentor and examiner: Prof. Gerald Q. Maguire Jr.

KTH Royal Institute of Technology
School of Information and Communication Technology
Stockholm, Sweden

Abstract

Sensors are light-weight, low powered devices that measure some aspect of a physical or virtual environment and transmit this information in some format. This thesis describes how to integrate a sensor onto devices to enable network connectivity.

The phrase “internet of things” suggests that within a few years many devices will be connected to an internet. Devices, including common household appliances, will transmit and receive data over a network. The CEO of Ericsson has stated that there will be more than 50 billion connected devices by 2020[1]. These devices could be microwaves, fridges, lights, or temperature sensors. Devices that are usually not associated with internet connectivity will be integrated into networks and play a larger role in providing information and controlling other devices. Sensors will have a major role in “the internet of things”. These small computers could be integrated in any appliances and transmit data over the network. The sensors’ low power and low cost, as well as their light weight, makes them very attractive to integrate them into many devices. The goal of this thesis project is to build upon this trend toward “The internet of things” by integrating a sensor into a bathroom scale thus enabling the scale to have networking connectivity. The sensor will be low cost and simple. It should wirelessly or via USB transmit the current weight that it measures to a receiver (specifically a gateway). This gateway will forward the message over the network to a website or mobile phone for visual presentation of the data. This thesis describes different techniques and approaches toward developing this sensor. The thesis also evaluates these different choices in order to select one technique that will be implemented. This solution will be evaluated in terms of its cost and ease of integration into an existing commercially produced scale.

Sammanfattning

Sensorer är små, energieffektiva apparater som upptäcker variationer i förhållande till någon stimulans och skickar informationen i ett godtyckligt format. Den här uppsatsen beskriver hur man kan integrera en sensor med en apparat för att möjliggöra nätverksuppkoppling.

Uttrycket ”The Internet of things”, översatt på svenska som ”Internet av saker”, konstaterar att inom några år så kommer den mesta av vardaglig elektronik vara uppkopplad mot Internet. Hemelektroniken kommer att skicka och ta emot data över ett nätverk. Ericsson tror att det kommer att vara över 50 miljarder uppkopplade apparater år 2020[1]. Sådana apparater kan vara mikrovågsugnar, frysar, lampor eller termometrar. Apparater som vanligtvis inte förväntas vara uppkopplade mot ett nätverk kommer att bli uppkopplade för att tillföra eller kontrollera information och andra apparater. Sensorer har en viktig roll i denna utveckling. De är små datorer som kan kopplas upp mot flertalet elektroniska apparater och förse dem med en nätverksuppkoppling för att skicka betydelsefull data över ett nätverk. Energieffektiviteten och de låga kostnaderna, så väl som deras små storlekar, gör dem dessutom väldigt attraktiva. Målet med den här uppsatsen är att påbygga ”The Internet of things” genom att tillförse en personvåg med en sensor för att möjliggöra nätverksuppkoppling. Sensor ska vara enkel och billig. Den kommer att trådlöst eller via USB skicka vikter till en mottagare som sedan skickar vidare informationen över nätverket till en hemsida eller mobiltelefon för en grafisk presentation av informationen. Den här rapporten beskriver dem olika teknikerna och approachen mot utvecklingen av sensorn.

Table of Contents

List of Figures

List of Tables

List of Listings

Glossary

ADC	Analog to Digital converter
Baud	Symbols/second
Comparator	Compares two voltages and indicates which is higher
CEPT	European Conference of Postal and Telecommunications Administrations
CoS	Department of Communication Systems, KTH, Kista, Sweden
CSMA	Carrier sense multiple access
D/A	Digital to Analog converter
ETSI	European Telecommunications Standards Institute
Gateway	A device that receives data from one system and transmits it onto another system
HW	Hardware
ISM	Industrial, Scientific, and Medical radio bands
LCD	Liquid Crystal Display is a type of display widely used in the computer industry
mA	Milliampere, one thousandth (10^{-3}) of an ampere
MHz	Megahertz
Op-amp	Operational amplifier. An electronic voltage amplifier with a differential input Produces an output much larger than the voltage difference between the two inputs
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PoE	Power over Ethernet, method to supply power to devices via the Ethernet interface port
SW	Software
RISC	Reduced Instruction Set Computer
USB	Universal Serial Bus, communication protocol used for communication and power supply between a computer and a device
VCP	Virtual COM port
µA	Microampere, one millionth (10^{6}) of an ampere

1 Introduction

The purpose of this bachelor's project is as the title suggests, "more than downloading"; to upload data to the internet. This was suggested by Prof. Gerald Q. Maguire Jr. who described the topic as follows:

> *"Today many networks are optimized for downloading (i.e., a transfer through the networking infrastructure to a host). Examples include web browsing, audio and video streaming from traditional media providers, etc. One of the coming trends is increasing generation of content (even if it is only a sensor saying "the temperature in the basement has fallen below 8°C"). This will result in increasing amounts of traffic in the uplink direction. [2]"*

As I had no or very little hardware experience, but have a large interest in hardware development and networking I found this topic to be very interesting for my thesis project. Working with hardware widens my knowledge, although it is a challenge as to how to adapt my software knowledge to the hardware part of computer science.

Many different sensors were considered (these are covered in Chapter 4) and a bathroom scale sensor was selected. The reason for choosing a bathroom scale is because of my interest in health and fitness. I believe that such a sensor would not only be fun to developed, but very useful as well. I am very confident that network connected bathroom scale at a low cost have a great market demand. This enables a wide range of possibilities for future work.

There are scales currently on the market that wirelessly connect to a network; these are discussed in Chapter 13. However there is an issue with these scales; they are expensive. The main goal with the sensor-project is to develop a low-end bathroom scale with connectivity possibilities. The costing of the scale has to be a minimal with no or little effect on quality. Another important goal is simplicity. The design and implementation have to very simple with no or very little requirement for user interaction. The user should only need to stand on the scale and a few moments later view the data on their mobile phone or a website. To achieve this in a realistic situation the purchaser should receive a gateway with the scale which has a built-in sensor and transmitter, or a wired solution via USB. From a sensor point of view the user should not have to do any configuration, however some configuration may have to be done for the gateway. Energy consumption is yet another important consideration. The sensor has to be low powered and synchronized with the scale's power consumption. The sensor's power should not be exhausted before the scale's power and both should have a similar averaged lifetime. Changing the power source should be inexpensive and simple, ideally not needed at all, for instance if piezoelectricity is sufficient to power the system (see Section 6.2).

My thesis project will consist of developing the sensor alone. It will used together with the gateway developed by Francisco Javier Sánchez and Albert López, described in section 2.3. The visual presentation and the application interface will be developed by Ivan Bremstedt Pedersen and Alfred Andersson for their thesis project which is described in Section 2.4. Figure 1-1 illustrates an overview of the system.

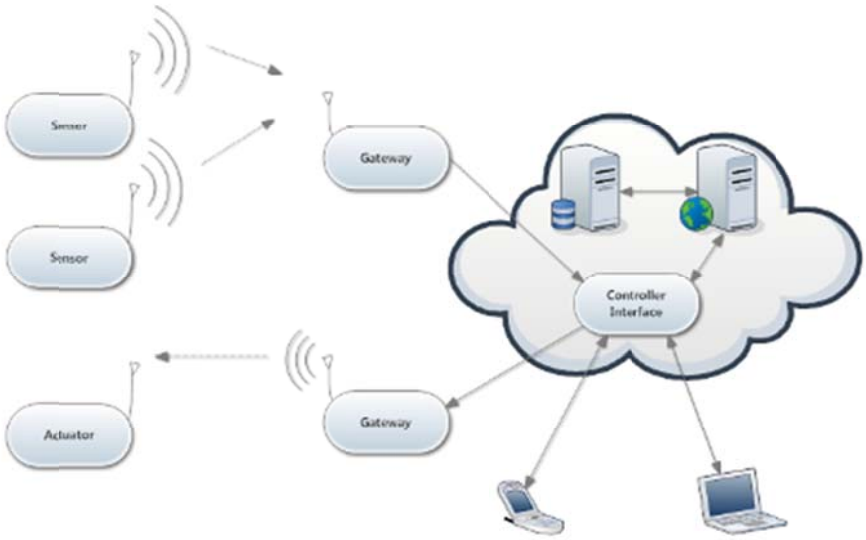

Figure 1-1: System Overview

1.1 Limitations of this thesis project

The initial idea of this project is to develop a simple scale sensor that transmits the data within a certain time frame. The sensor will send a simple frame with an identification field and the weight and measurement unit as data, no other information will be included. Such information could be date and time, temperature, the number of clothing or other factors that might be interesting in regard to weight. The sensor will also disregard the user, it will not be possible to distinguish between the people using the scale. It is assumed that the scale will have a unique user. It will also not be possible to connect the scale to any network and send data through it in the case it is a wireless solution. The sensor would be dependent on one gateway. Furthermore, the user will not have the option to not include a weight (unless they turn off the sensor or gateway), every weight will be transmitted. The sensor will also most likely be time sensitive, for instance it will be required of the user to stand on the scale until the weight measurement has stabilized. If the user steps on and off the scale the transmitted weight will be incorrect. In this project the sensor will not be built in to the scale, it will be an external device.

1.2 Thesis structure

Following this introduction, Chapter 2 will describe related and similar work done by other. Chapter 3 introduces the Wasa board used in this project and Chapter 4 will describe different types of sensors that might be used. Chapter 5 describes some of the wireless physical and link layer technologies that might be used to connect the sensors to a gateway, while Chapter 6 describes experiments that I have made to augment a scale, enabling it to be a network connected sensor. Descriptions of the MSP430 microcontroller and a brief programming guide are provided in Chapter 7. Chapter 8 describes the initial experiments with the Wasa board. Chapter 9 describes the "Hello World" software implemented on the MSP430 and chapter 10 discusses the first weight decoding algorithm. A second algorithm with the use of AT commands is described in Chapter 11. Chapter 12 describes the program that saves the weights locally to a file. A market research looking into other scales on the market as well as the cost of this thesis project is discussed in Chapter 13. Chapter 14 discusses future work, while Chapter 15 and Chapter 16 discuss the results and conclusions of this thesis project.

2 Related work

After some research on the internet I found that other people had done similar work with a bathroom scale. Jorge Pinto connected sensors to a couple of different scales and used the data for different purposes. One scale wrote data to an SD card, while another transmitted the weight via a Bluetooth link to a computer. Jorge Pinto's projects [3] are described in more detail in section 2.1. Another similar project by Micah Elizabeth Scott [4] is described in section 2.2. Her project is very interesting as she reads the analogue signals directly from the sensors ignoring the microcontroller and the LCD screen.

Section 2.3 describes Francisco Javier Sánchez and Albert López thesis project where they built the gateway which I may be using. Furthermore describes section 2.4 the application interface that will be developed by Ivan Bremstedt Pedersen and Alfred Andersson for their thesis project.

2.1 Jorge Pinto – SD card

Jorge Pinto has done 3 different projects with different scales. He started out with the Fagor BB-90 scale, a cheap and simple scale where he snooped the data from the scale's LCD screen. The goal of his project was to store the data on an SD card and further add more advanced features to the scale. He did so by removing the original LCD screen and then added his own LCD screen and a microcontroller (LPC2103) which he could easily control. The microcontroller fetched data from the SD card and did multiple calculations before outputting it on the LCD screen. The new features were time and date, increase or decrease in weight, and multiuser support. This scale was further developed to give it Bluetooth connectivity. The scale communicated with his Android mobile phone via a Bluetooth link. He uses the Android application **Smart Weight Chart** to show the data as a graph [5].

His second project was similar to the first one, but with a different scale (Jata Hogar 490). This project is very interesting as the LCD screen works similarly to the LCD screen on my scale (see Chapter 6). Understanding his project was of great help when decoding my own scale.

Thirdly, he manipulated the Beurer BG 16 scale which measures weight, percentage of fat, percentage of water, and percentage of muscle. He made the same modifications as with the other scales.

2.2 Micah Elizabeth Scott – hacking a digital scale

Micah Elizabeth Scott's project is very interesting as she does not fetch the data from the LCD screen, but instead connected directly to the sensors. Beth traced where the signals from the sensors went and figured out the analogue front-end then simply connected to it and passed these signals to a microcontroller (Propeller). This decreased the number of required pins on the microcontroller as only two I/O pins were required. She samples the weight signal by keeping an integration capacitor balanced around a comparator threshold. By measuring the amount of time it takes to charge the capacitor the weight upon the scale is calculated. Her work could be very interesting for another student to improve upon my work.

2.3 Sánchez and López (Master thesis project)

A gateway is needed to relay the information received via a radio signal to a computer connected to a network. This gateway contains a radio receiver that listens to a radio channel at a specific frequency. The gateway sends the information to a local computer via USB or via an Ethernet controller. In this thesis project I might use the gateway developed by Sánchez and López [6].

Francisco Javier Sánchez and Albert López have during the winter/spring 2012 developed a gateway that uses the 868 MHz band to snoop sensor data and send this data to a computer via an Ethernet interface. Their gateway uses a MSP430 microcontroller and a TI CC1101 wireless transceiver to communicate with different types of sensors. This gateway requires very low power and

is able to operate on power over Ethernet (PoE) or from an external power supply. Their gateway contains only low cost components. See their thesis report for a detailed description [6].

2.4 Pedersen and Andersson (Bachelor's thesis project)

Ivan Bremstedt Pedersen and Alfred Andersson are developing the application interface for viewing sensor data. Their goal is to modify the gateway developed by Francisco and Albert to send data securely to a web-server for visual representation on a website. Another goal is to implement a TCP/IP-stack in the gateway to be able to communicate with the web-sever without requiring an additional computer.

A web interface was developed. This interface consists of a webpage running HTML5 and JavaScript code, together with Ruby on Rails in the background for authorization and login (in order to control who can view data collected from a sensor). Many different possibilities have been discussed, for instance mashing up with Facebook, either for logging in or possibly show a chart of the weights in Facebook.

A mobile application has also been discussed, similar to the project by Pinto described in section 2.1. A web application would serve the same purpose as a website, but could add mobility to the application interface. See their thesis report for a detailed description of their application interface [7].

3 Wasa Board

For this thesis project I used the Wasa controller board. The Wasa board was designed by Professor Mark T. Smith to enable students to easily add sensors to existing computing platforms, such as a laptop or PDA. The Wasa board contains a MSP430F2618 microcontroller, a FT232RL USB client controller, a temperature sensor, a light sensor, and a 3-axis accelerometer. Details of these sensors are described in Chapter 4 and summarized in Table 3-1. The initial software on the WASA board uses AT-commands for communication, hence the Wasa board accepts commands starting with 'AT'. It's possible to directly communicate with the Wasa board using command-prompt terminals via serial ports.

The front and backsides of the Wasa board (version 1.7) are shown in Figure 3-1 and Figure 3-2. Together with Alfred Andersson and Ivan Bremstedt Pedersen have we used the board for experiments to gain an understanding about these different sensors. The Wasa board also enabled us to prototype our ideas of several possible sensor and actuator scenarios without having to build new hardware. A number of experiments with the Wasa board are described in Chapter 8.

Table 3-1: Summary of Wasa board version 1.7 components

Part	Vendor	Description
MSP430F2618	TI	Microcontroller
FT232RL	Future Technology Devices International Ltd.	USB client controller
Thermistor NTC 10K 5%	Vishay BC Components	Temperature sensor
MPY20C48	Silicon Sensors	Light sensor, and
MMA7660FC I2C.	Freescale	3-axis accelerometer

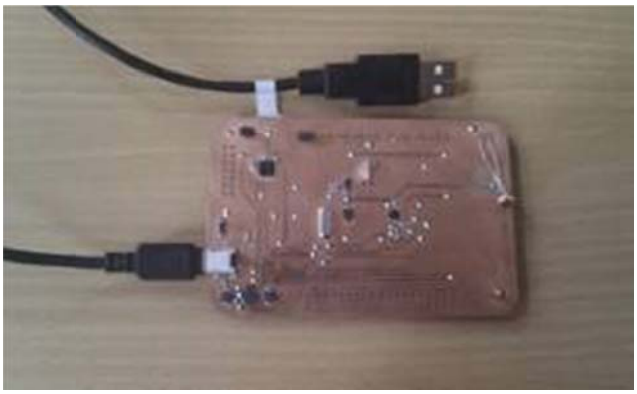

Figure 3-1: One side of the Wasa board (version 1.7)

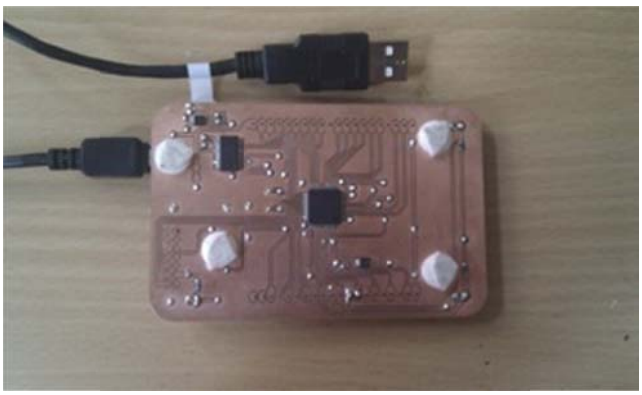

Figure 3-2: Other side of the Wasa board (version 1.7)

4 Sensors

A general goal of sensors is to collect as much information as possible in order to extend a system's functionality. With sensors for light level, air temperature, humidity, accelerometer, or motion sensors a system would have endless uses. There are a many different sensors that could be of interest. This chapter describes the different sensors that were considered during the literature study.

4.1 Light sensors

The Wasa board, described in Chapter 3, has a light sensor MPY20C48, produced by Silicon Sensors. The initial tests with the light sensor on the Wasa board were successful. These tests are described in section 8 (starting on page 23). I was able to see a difference in the sensor's value by simply placing a hand over the light sensor.

4.2 3-axis accelerometers

The Wasa board also has an accelerometer, a Freescale MMA7660FC I2C. The initial tests showed that I was able to read three values from the accelerometer. Each output consists of three values for X, Y, and Z. The test I conducted is described further in section 8 (starting on page 23).

4.3 Air sensors

Sensors that collect data about the air could be useful for homes. With data from these sensors users could start the air conditioner, turn up or down the thermostat, open or close the window blinds, and so forth. These functions are often mentioned when people talk about the "smart home". Additionally, this data could be used together to lower their electricity bills and help reduce the user's carbon footprint.

The two sensors described below could easily be incorporated because they are so closely related. Placing them in the same device will save money because only one transmitter and microcontroller is needed for the two sensors.

4.3.1 Temperature sensors

The Wasa board uses a temperature sensor that I have been able to take readings from. The test method and results are shown in section 8 (starting on page 23).

4.3.2 Humidity sensors

Humidity sensors could also be used with the Wasa board. I have at this time not conducted any experiments on how to use these sensors. An example project that utilizes a humidity sensor can be found on page 105 in Thomas Petruzzelli's book "*Electronic Sensors for the Evil Genius"* [8].

4.3.3 Possible usage scenarios

These different types of sensors could be placed in each room of a home in order to enable the user to monitor each room and in turn control the local environment with potentially individual settings (not only as a function of the date and time, but also whether people were in the room or not). They could also be placed in a garage or basement where the air's humidity could be of concern to the user. For example, an increase in water vapor in the air might be a sign of a leak, if the humidity has not also increased outside of the home.

If the device is placed in a waterproof case, then an outdoor installation is also possible. The lab's 3D printer could print such a case.

Another possible usage scenario is a sauna where the electrical power or water supply to the heater could be controlled by the system via sensors placed in the sauna. Due to extreme temperature and humidity in a sauna special care needs to be taken to insure that the components function correctly.

4.4 Weight sensor

A weight sensor that collects data about the weight of a person would allow the user to monitor her/his weight. A scale (Coline RTC3010) has been purchased from Clas Ohlson (SKU: 34-5062) and initial tests describing how to read data from this scale are in section 6.3 (starting on page 11).

The scale sensor will communicate with the gateway developed by Sánchez and López as explained earlier or send data via USB. In the case a wireless solution is implemented, it will transmit in the 868 Mhz band, most likely at 868.26Mhz, the same frequency as the temperature sensor they are currently working with. Ideally the sensor would transmit using the ZigBee protocol because the signals will be transmitted using CSMA techniques which are highly desired. Also as the ZigBee offers a payload of 104 bytes it would be more than enough to transmit a weight in a single packet. However the ZigBee stack is closed and requires a license. Therefore the packets could be transmitted using a proprietary protocol, the SimpliciTI. The data will be transmitted multiple times for the duration the scale stays active to minimize the risk of undelivered packages. The frame will most likely be very simple with only an identifier and data fields. If there is time a cyclic redundancy check (CRC) code will be included as well. The weight will be binary-coded decimals (BCD) and the signal will be Frequency shift keying (FSK) modulated. ZigBee, SimpliciTI, and other wireless transmission techniques are covered in Chapter 5.

5 Wireless transmission technologies

One of the main goals of this project is to communicate wirelessly between different devices. This wireless communication has to follow conventions and rules that are determined by international and national regulations (CEPT and ETSI). There are restrictions on which bands that can be used without a license. These bands are ISM-bands and one of the frequency bands that can be used in Europe (Sweden) is the 868MHz band. The 868MHz frequency is especially interesting since I may be using the gateway built by Francisco Javier Sánchez and Albert López as described in section 2.3, which operates in the 868MHz band.

Section 5.1 describes the different IEEE 802.15 wireless transmission technologies as well as the SimpliciTI protocol. Section 5.2 briefly describes the radio transmitters that have been discussed.

5.1 IEEE 802.15 and SimpliciTI

There are many different wireless transmission technologies each suitable for a particular need depending on data rates, communication range, security, and power consumption among other factors. Local area networks, wide area networks, metropolitan area networks, cellular wireless networks, and satellite communications each serve a purpose depending on the factors mentioned above. For this thesis project Wireless Personal Area Networks (WPAN) are of most interest. WPANs are wireless local networks for short range devices operating in the ISM-bands. The IEEE 802.15 Working Group for WPANs was formed to developed standards for short range wireless communication [9]. The first standard that was developed was IEEE 802.15.1 which was based on Bluetooth technology. Following the IEEE 802.15.1 standard the working group deployed further task groups. One of the task groups IEEE 802.15.3 was interested in developing standards for devices that are low cost and low power but with higher data rates than IEEE 802.15.1. The IEEE 802.15.4 working group is less concerned with data rates and thus develops standards for devices that are low cost and low power but with data rates lower than 802.15.1. My main concern lies with low power and low cost and as the packet sent from the scale sensor will most likely be small therefore a high data rate is not needed; hence the IEEE 802.15.4 standard is very interesting. Table 5-1 summarizes these standards in terms of power, operating frequency, range, and data rates.

ZigBee is a protocol that runs on top of IEEE 802.15.4. The ZigBee protocol is widely used in many different communication systems because of its ease of implementation. Because it operates on top of the IEEE 802.15.4 standard the ZigBee protocol can keep low power consumption. The purpose of the ZigBee protocol is to provide a standardized base set of solutions for sensors and control systems. However, since the ZigBee protocol stack must be licensed, other proprietary protocols have been considered. One of these protocols is SimpliciTI [10]. SimpliciTI is a simple low-power radio frequency (RF) protocol for small RF networks developed by Texas Instruments (TI). It is suitable for devices that require long battery life, low data rates, and low duty cycle. The scale sensor is such a device. It was designed for easy implementation on platforms such as the MSP430 microcontroller and the CC1101 radio transceiver. SimpliciTI will most likely be used for transmitting the weights to the gateway.

Table 5-1: Summary of 802.15 network technologies

	Bluetooth	UWB	ZigBee
Standard	IEEE 802.15.1	IEEE 802.15.3	IEEE 802.15.4
Transmit Power	Low	Low	Very Low
RF Band	2.4 GHz	3.1-10.6 GHz	868/915Mhz, 2.4GHz
Range	10m	30m	30m
Peak Data Rate	1Mbps	1.6Gbps	250kbps

5.2 Radio transmitter or transceiver

In order to be able to send information wirelessly from each sensor to a central receiver or receivers, a transmitter or a transceiver is needed. A transceiver is simply a combined receiver and transmitter. Using a transceiver would allow full duplex communication with sensors or actuators. I have considered two low cost radios. These are described in the following two subsections.

5.2.1 RFM02 - transmitter

The Hope Microelectronics Co., Ltd. RFM02 is a low power, single chip radio transmitter. It is capable of transmitting in the 433, 868, and 915 MHz ISM bands. Details of this transmitter module are available at *http://www.hoperf.com/rf_fsk/fsk/19.htm*.

5.2.2 Texas Instruments CC1101 - transceiver

Francisco Javier Sánchez and Albert López have used the TI CC 1101 [29] in their master's thesis [3] in order to receive and transmit in the 868 MHz band. I can more easily build upon their work if I use the same components. Because this device is a transceiver it can be used to both receive and transmit, providing great flexibility and supporting many possible usage scenarios.

The CC1101 operates at sub 1 GHz frequencies[1] and is used for very low-power applications. The circuit also has a low-cost. It has high sensitivity (-112 dBm) at 1.2kBaud (868MHz), low current consumption (14.7mA) while receiving, and a programmable data rate from 0.6 to 600 kbps.

[1] Typically in the 315, 433, 868, and 915 MHz ISM bands; often the ranges: 300-348 MHz, 387-464 MHz and 779-928 MHz. For further details see http://www.ti.com/product/cc1101.

6 Scale

My main focus is a bathroom scale (i.e., a weight sensor) that would collect weight data and wirelessly transmit it to a gateway or via USB. The information would pass over to a server and could be accessed through a web browser or mobile phone. A visualization of the data in form of charts (weight over time) could be shown.

The inner working of a scale is described in section 6.1 and section 6.2 discusses the different power sources that have been considered during the project. Section 6.3 describes the number of different approaches to understand and read data from the scale.

6.1 Inside a digital scale

To be able to read data from the scale a general understanding of a scale's inner working was necessary. A low cost digital scale makes use of strain gauge technology, i.e., the measurement of strain due to the weight of an object. The electrical resistance of this strain gauge changes as the weight of the object deforms the strain gauge.[11] In the case of a scale the object that stretches causes an increase in electrical resistance. In a scale the bending objects are load cells usually in the form of a bar. Load cells in a scale are transducers that convert the force to analog signals. In a well-designed scale the weight is evenly distributed over one or two load cells. The weight is applied to one end of the load cell causing a change in the electrical resistance. The strain gauge sensors utilize the change in electrical resistance to calculate the applied force. This force is then converted to an analogue signal that is passed through an ADC and calculations are done in the microcontroller and the weight is displayed on an LCD screen.

6.2 Sensor Power

The sensor will most likely be powered over USB at first; however, there is a question of how the sensor should be powered at a later stage, especially considering if it has to power a wireless transmitter. Table 6-1 shows the three different options that have been considered. Perhaps is a combination of two different power sources the ideal solution.

Table 6-1: Option to power the microcontroller and radio transmitter

External AA battery pack	The sensor could be powered by an external battery pack with AA batteries. This would provide independent power, sufficient for a long operating time. However, it will increase the cost of the device and clumsiness. The battery pack would require space, which is not always available within the scale. If there is space, then the battery pack would be installed inside the scale, but it still will be an increase in cost.
Scale's batteries	Instead of using an external battery pack the sensor could make use of the scale's internal batteries. The disadvantage of this approach is that the sensor will drain the scale's batteries much quicker than normal. However, some analysis of the cost efficiency and battery lifetime has to done to conclude whether this approach could be beneficial regardless its disadvantages. The scale that will be used is the Coline RTC3010 that uses one CR2023 battery. The CR2023 battery is a coin cell lithium battery for low power applications. During testing that consisted of connecting the LCD screen pins to an oscilloscope for a couple of hours, the battery was drained. This suggests that using the scale's battery is not a feasible means to power the system.
Energy harvesting	The system might possibly be powered through harvested energy using piezoelectricity. Piezoelectricity is electrical energy produced from mechanical pressure. [12] It is quite similar to the strain gauge technology. Electrical energy would be produced when the scale is stepped on. The scale could be outfitted with piezoelectric discs and each time pressure is applied to the scale the mechanical pressure would be converted into an electrical charge by the piezoelectric discs and stored (most likely in capacitors) and used as a power source. The question is how much energy is produced and how much energy the system needs.

6.3 Reading weight data

There are different approaches to get the data from the scale. One approach is to snoop the calculated data going to the LCD screen. Another approach would be snooping the voltages from the scale's sensors and independently calculating the weight. However, each of these approaches raises questions. Is the Wasa board's ADC accurate enough? Will snooping the data to the LCD screen interfere with the LCD screen's display? The LCD screen is most likely in different states at different times, at what point should the data be snooped? Or should multiple samples be taken? This section describes a number of initial tests of reading the weight data that were made with a Coline RTC3010 scale.

The RTC3010 is Clas Ohlson's lower end, digital, simple scale (SKU: 34-5062). It is made out one glass brick and four rubber feet with a compartment at the header of the glass brick for the LCD screen and the electronics. The scale has no advanced features other than the selection of different measurement units: kilogram (kg), pound (lb), or stone (st). Thus this scale does not include features such as inputting and displaying the user's name or storage of previous weights. The simplicity of the scale enables it to be low power and thus it has only one CR2023 battery. The front and back of this scale are shown in Figure 6-1 and Figure 6-2 (respectively).

Figure 6-1: Front view of the scale

Figure 6-2: Back view of the scale

Instead of having one or two larger load cells the scale has a simple design with four small load cells, one at each foot (See Figure 6-3). Each load cell uses strain gauge sensors connected to the printed circuit board (PCB). The force caused by the user standing on the scale causes a change in electrical resistance and this change is converted into an analogue signal. The signals from the load cells are combined and converted into a weight that is displayed on the LCD screen.

Figure 6-3: Electronics overview

6.3.1 Different Approaches to snoop the data

In this subsection I describe my attempts to snoop the data from the scale's existing electronics to the LCD screen.

6.3.1.1 *Voltage sensing*

The first approach was to snoop the voltages directly from the sensors. This requires some reverse engineering of the scale and calibration to map the different voltage levels to different weights. The first step was to solder additional wires to the connections from the sensors to the PCB. The additional wires were connected to an oscilloscope to look for the differences in output for a small number of different weights that I applied to the scale. These readings could enable us to calculate the corresponding weight. However, this approach did not work as the readings did not change; I saw only noise regardless of the force applied to the scale. A different method that might have given a better result would have been to solder the additional wires straight to the sensors. In this approach the sensors could be directly read. However, due to the scale's design, each sensor is covered by a metal bar, hence it was not possible to reach the appropriate points on the sensors without having to make a major change to the physical scale.

6.3.1.2 *LCD screen*

Fortunately the scale's design provided other possibilities. The LCD screen's pins were available on both sides of the PCB. This made it possible to read the data going to the LCD screen *without* interfering with the screen itself. A front and a back view of the PCB are shown in Figure 6-4 and Figure 6-5 respectively.

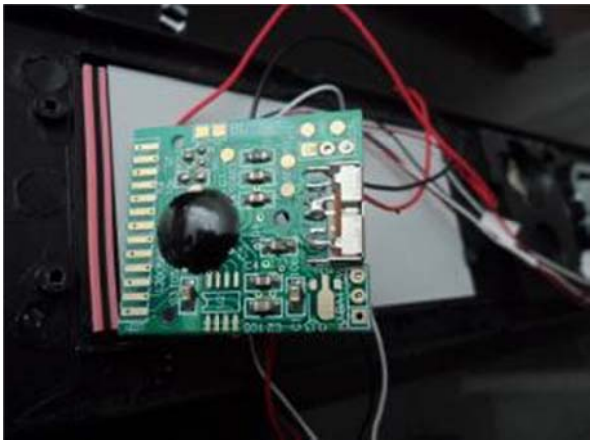

Figure 6-4: Front view of the scale PCB

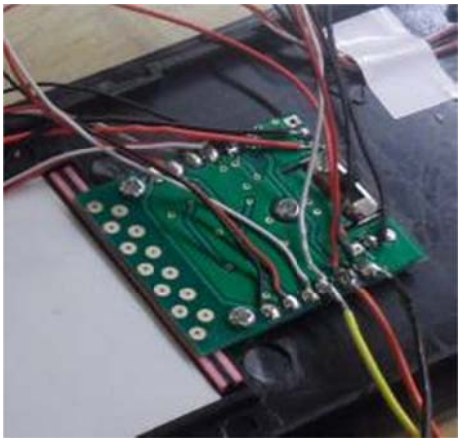

Figure 6-5: Back view of the scale PCB

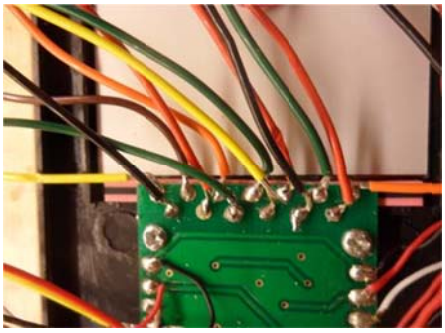

Figure 6-6: soldered pins

The initial idea was to snoop the signals going to the LCD screen. There are 13 pins in total going to the LCD screen, thus a wire was soldered onto each pin, see Figure 6-6. The pins were numbered in a zigzag fashion where the first pin was the rightmost pin (as shown in Figure 6-5).

After some research on the LCD screen I was able to understand that each number was represented as a seven-segment digit, as shown in Figure 6-7. The LCD screen consists of four digits, one decimal point (DP), and three measurement units

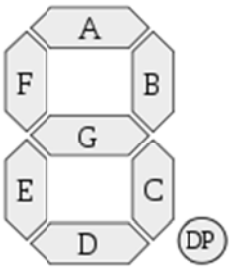

Figure 6-7: Seven-segment digit [4]

The first attempt of reading the weight data was to find out which pins control which segments. A reading was done on each pin at an applied weight of 0.0 kg. This was done by observing the different signal patterns (see Figure 6-8 and Figure 6-9) at each pin and in order figure out how they correspond to each other and the seven-segment digit. This had me believe that the analogue signals were constructed as bit patterns where each change in phase in the signal corresponded to a bit, 1 or 0 that determined the segment's state. Without much understanding of the patterns themselves I was still able to map different patterns to different digits and segments. Each pin was later on recursively tested for different digits.

Through extensive testing and observation was I able to come to the conclusion that each pin is responsible for 4 segments, thus 2 pins were responsible for a digit. I also came to the conclusion that the first pin (rightmost pin in Figure 6-5) was responsible for the measurement unit. The least significant pin for a digit was responsible for the x-c-b-a (x being the decimal point) segments and the most significant pin was responsible for the d-e-f-g segments, see Figure 6-7. However, I was not able to understand the purpose of the last 4 pins as they connected no segments on the LCD screen.

There was still some confusion after the initial attempt of understanding the scale. For instance the purpose of the last 4 pins was unclear and the bit order was not understood. Furthermore I did not understand the patterns and how and why they changed as they did, see Figure 6-8 and Figure 6-9. As a result some extensive research was carried out and a second attempt to understand the scale was made. The second attempt is described in section 6.3.2.

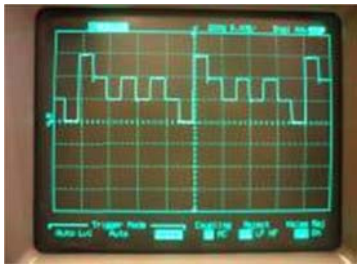

Figure 6-8: Oscilloscope readings of the 1st pin

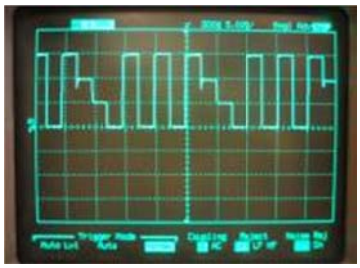

Figure 6-9: Oscilloscope readings of the 2nd pin

6.3.2 Second Test

An LCD is driven as a matrix of signals and so is the scale's LCD display. Each signal as shown in the first test is by itself meaningless. However the intersection of two signals determines whether a segment is on or off. Every LCD is driven by one or many backplane signals, as explained in section 6.3.2.1. An LCD has also several segment signals, each signal could represent multiple segments or there could be dedicated signal for each segment. The backplane signals are either the vertical or horizontal lines in the matrix and the segment signals are the lines opposite in the matrix. The intersection of a backplane signal and a segment signal determines whether that particular segment is on/off, illustrated in Figure 6-10. A segment is illuminated when both a vertical and a horizontal line are suitably connected to power and ground. The signals are continuous and driven at a certain frequency which to the eyes seems like all the segments (for a particular digit) are always ON, but technically only the segments intersecting with the active backplane are ON, when there are multiple backplanes. Any direct current (DC) damages the display [13].

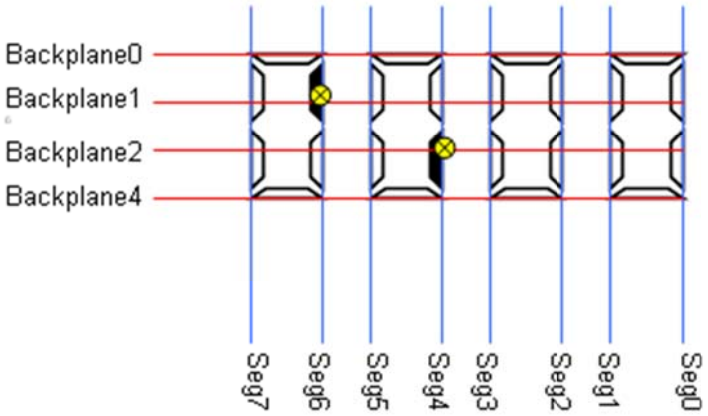

Figure 6-10: How an LCD is constructed as a matrix

6.3.2.1 *Types of LCDs*

There are two types of LCD screens: static LCDs and multiplexed LCDs. They are described in the following sections.

6.3.2.1.1 Static LCD

Static LCDs have one backplane line (pin) and a dedicated line for each segment of the LCD. The LCD is driven through the input of a square wave. To turn a segment 'on' the corresponding line has to be in opposition to the backplane signal. To turn a segment 'off' the segment signal has to be of the

same phase as the backplane signal, as shown in Figure 6-11. The difference between the signals at each phase determines then if the segment is on or off. If the difference is 0v, then the segment is off while if the difference is non-zero voltage, for instance 3v the segment is on. This design is simple but as each segment requires a dedicated pin, a larger number of pins and connections as well as a larger PCB are required. The microcontroller also needs more output/input pins.

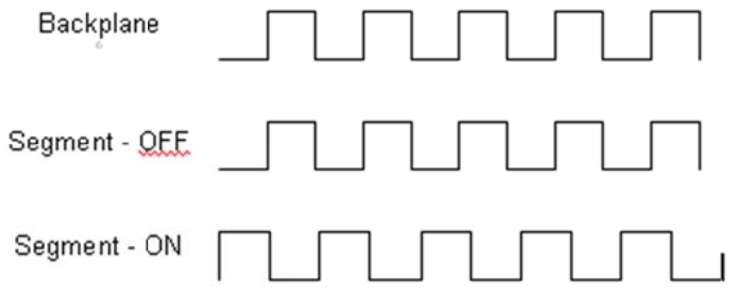

Figure 6-11: static LCD

6.3.2.1.2 Multiplexed LCD

A multiplexed LCD has several backplane signals and a single segment pin is shared among multiple segments. To turn 'on' a segment the responsible signal is driven in opposition to the active backplane signal while the remaining backplane signals are idle during that particular duty cycle as illustrated in Figure 6-12. A duty cycle is the period one backplane stays active, the number of backplanes determines the number of duty cycles during one frame. A wave is a continuous sequence of frames. For a segment to be visible the difference between the driven segment signal and the backplane signal has to above a threshold voltage which is usually ~2.3V. If the difference between the voltages is above the threshold the segment is turned on, otherwise it is off. To have an average zero DC bias (mean value of a signal) the difference between a backplane signal and a segment signal during a frame has to total out to zero voltage as shown in Figure 6-13[14]. Thus the backplane signals idle at mid-voltage, see Figure 6-12.

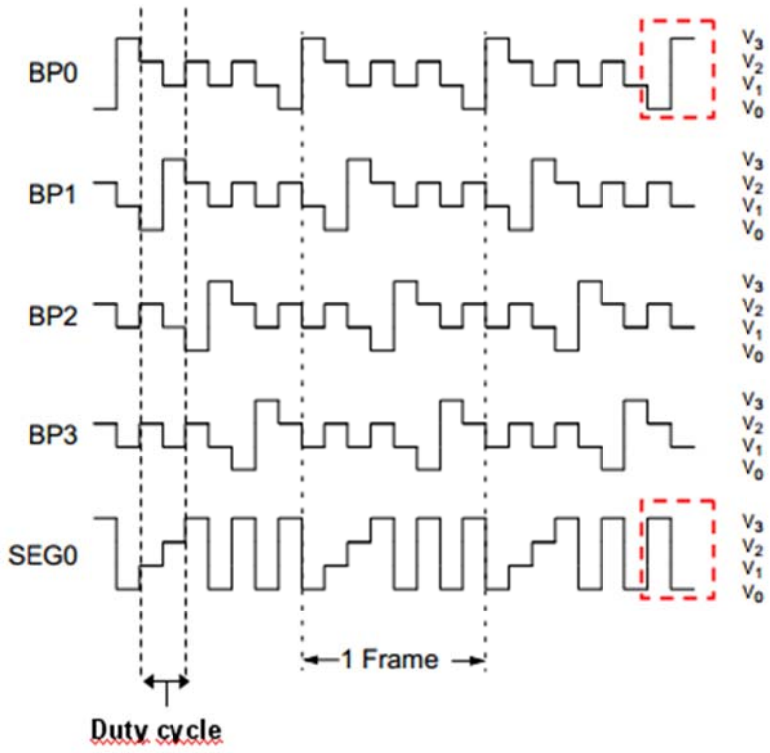

Figure 6-12: Multiplexed LCD [13]

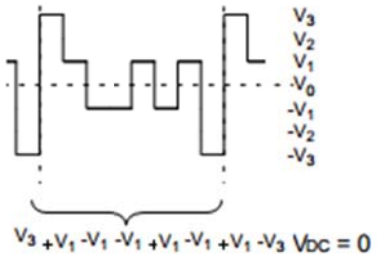

Figure 6-13: Zero DC bias within a frame

6.3.2.2 *The scale's LCD screen*

The scale's LCD screen is a multiplexed LCD with 4 backplane signals. Each backplane signal is responsible for 2 segments as shown in Figure 6-14. As previously found in the initial test 2 signals (pins) hold each digit; that is each signal is responsible for 4 segments (see Figure 6-15). One signal drives the x-c-b-a segments and the other signal drives the d-e-g-f segments. Table 6-2 shows what segment each intersection between a backplane line and a segment line is responsible for. I came to these conclusions in the first test, but to confirm these conclusions with a lot more background knowledge in the second test I read two signals at once, one backplane and one segment signal. If the segment intersected between the backplane signal and the segment signal is on then the two signals should be opposite (power and ground) during that phase. This is illustrated in Figure 6-16.

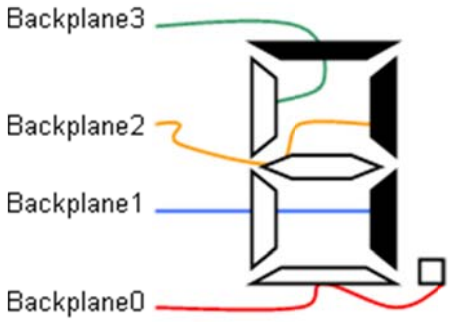

Figure 6-14: Horizontal backplane signals

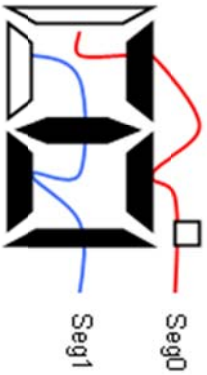

Figure 6-15: Vertical segment signals

Table 6-2: Segments at each intersection between a backplane line and a segment line

	Segment line 0	Segment line 1
Backplane 0	DP	d
Backplane 1	c	e
Backplane 2	b	g
Backplane 3	a	f

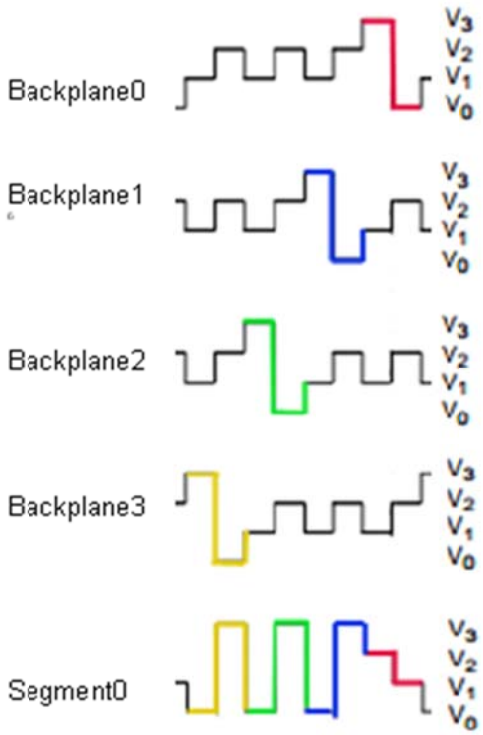

Figure 6-16: 2nd pin and backplane signals at the value of 0

Figure 6-16 illustrates the least significant digit pin which holds 4 segments (x-c-b-a) at the value 0. As described in section 8.4.2.1.2 each segment is illuminated during one duty cycle. At the first phase (red) the signals are not in opposition (power resp. ground), thus that segment is off. The first backplane signal and the least significant digit pin intersects at the segment x, see Figure 6-14 and Figure 6-15. The second-fourth backplane signals intersect with the segment signal at the segments c-b-a and as the segment signal is in opposition to each backplane at each corresponding phase those segments are on.

The same observations were done for different digits of the LCD display. At the intersection of two signals that segment is on if the signals are in opposition, hence connected to power and ground respectively.

7 Understanding MSP430

Having figured out the technical details of the scale it was time to familiarize myself with the hardware that is going be used in this project, the Wasa board and the MSP430 microcontroller. The Wasa board is discussed in more detail in Chapter 8, while this chapter covers the MSP430 microcontroller. A brief description of the MSP430 is provided in section 7.1. Section 7.2 focuses on programming. There are courses covering MSP430 programming as it is a broad topic, thus only the relevant parts of programming this chip, such as the ADC, will be briefly discussed in this chapter.

7.1 Texas Instruments MSP430

Prof. Maguire recommended the MSP430 microcontroller as it has been used in several other recent thesis, including the bachelor thesis of Thor Hådén [15]. The TI MSP430 is a low voltage and low power microcontroller that can be used to collect data from sensors, control an actuator, and send and receive data via a radio. It's a 16-bit RISC microcontroller, where the "16-bit" means that all registers hold 16 bits. The reduced instruction set computer (RISC) acronym means that the controller has a reduced number of core instructions – in this case "only" 27 of them. The MSP430 is a family of controllers including the one used by Hådén: the MSP430F2618. This particular microcontroller is also used on the Wasa board (described in Chapter 3) that I have been experimenting with. The MSP430 operates on a low supply voltage ranging between 1.8 V and 3.6 V. It features a built in 8 channel, 12 bit analogue-to-digital (ADC) and two digital-to-analogue (D/A) converters.The specific MSP430 that I have used is the MSP430F2618. This version of the chip contains 116KB + 256B of Flash Memory and 8KB of random access memory (RAM).

7.2 Programming the MSP430

Understanding the programming syntax of the MSP430 is the most crucial part of this project. This is also where most of the time was spent. A microcontroller is a small computer with a processor, memory and I/O meaning that as a programmer I need to consider all of these aspects. Programming at a low level, for embedded systems could be more difficult than traditional high level programming as many more things needs to be considering, such as I/O, interrupts and clocks. This section describes the steps taken from the very basic understanding of binary to the more complicated understanding of the analogue-digital converter (ADC) and registers. This may be of great help to future students.

7.2.1 Data types and registers

The MSP430 is as most other computers a binary computer. Meaning they operate on '1' and '0'. Luckily enough as a programmer I do not need to concern myself with binaries as it may get a bit tricky, the reason is that the compiler sorts out the chosen numeric presentation to binary. The most used numeric presentations are binary, hexadecimal and decimal, in most cases hexadecimal; therefore a clear understanding of hexadecimals is required. Hexadecimal uses 16 symbols to represent numbers, 0-9 followed by A-F, whereof F equals 15 in decimal or 1111 in binary as shown in Table 7-1. Using hexadecimal decreases the number of symbols needed to represent a number. Each symbol in hexadecimal represents 4 bits, enabling one symbol to represent 16 decimal numbers and two symbols to represent 256 decimal numbers. Hexadecimals numbers are denoted with *0x* followed by a number of symbols.

Table 7-1: Numeric presentations

Hexadecimal	Decimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111
0x10	16	10000
0x11	17	10001
0x12	18	10010
0x13	19	10011
.	.	
.	.	
.	.	
0x1F	31	11111
0x20	32	100000

It is very important to understand that even though hexadecimal notation is used, the actual arithmetic takes place in binary. MSP430 programming relies heavily on registers and binary operations. Thus a good understanding of bitwise operations is very handy. The most frequently used bitwise operations are AND, OR, NOT, and XOR, see Table 7-2 for their truth tables.

One very common MSP430 programming example is the blinking of a LED, to achieve this, the LED's value (GPIO output) will XOR itself continuously, shifting from 1 and 0 (on and off). The following pseudo code gives a brief clarification;

```
For (;;)
{
LED_Port_0 ^= 0x01;
delay(1sec);
}
```

Listing 7-1: Blinking led – XOR example

The LED is turned on and off in a continuous loop every second. I assume that LED_Port is associated with a register with 8 bits connected to 8 pins, 1 bit per pin. I also assume that the LED is connected to the first pin (pin 0). Setting this bit to 1 turns the pin to high, hence 0x01 (0000 0001). As shown in Table 7-2 XOR 1 with 1 return 0 and XOR 1 with 0 returns 1 resulting in turning the LED on and off by XOR the same value with the current contents of the register.

Table 7-2: Logical Operations – OR, AND, NOT and XOR [16]

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

(a) OR, |=

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

(b) AND, &=

A	$\tilde{A}$
0	1
1	0

(c) NOT, !=

A	B	A ∧ B
0	0	0
0	1	1
1	0	1
1	1	0

(d) XOR, ^=

Having refreshed my memory on bitwise operations it was time to understand the many, very important registers in the MSP430. The registers that were of interest are the module registers that control I/O. These registers could be 8 or 16 bits in width. Setting a bit of a register is done using the bitwise operations discussed above.

An interesting and important register in this project is the ADC control register ADC12CTL0. Figure 7-1, found in the Texas Instruments, 'MSP430x2xx Family - User Guide' [17], shows each bit in the register. For instance setting bit 0 of ADC12SC to 1 starts the ADC conversion, while setting it to 0 stops the conversion. See this User Guide for further detailed explanations of the different bits of this register and the many other registers. To simplify programming the different bits can be set using their assigned names, as an example:

ADC12CTL0 |= ADC12SC;

is the same as starting the conversion by saying:

ADC12CTL0 |= 0x0001;

15	14	13	12	11	10	9	8
SHT1x				SHT0x			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
MSC	REF2_5V	REFON	ADC120N	ADC12OVIE	ADC12TOVIE	ENC	ADC12SC
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Can be modified only when ENC = 0

Figure 7-1: ADC control register - ADC12CTL0

7.2.2 ADC and GPIO

The most important aspect of this project is the ADC. The analogue signals from the scale's LCD display are to be inputs to the ADC. The ADC will sample these signals and return a value representing the voltage at the sampling time. Converting the analogue voltage to a specific digital value is known as *Quantization* and the returned value is based upon a quantization level. Each level represents a specific voltage threshold. The ADC's resolution determines how many levels can be used to represent a signal; thus dictating the lowest difference in input voltage we can differentiate [16], the more levels, the higher the accuracy of the sampling. The MSP430F2168 used in this project has an ADC resolution of 12-bits, thus the number of quantization levels are $2^{12} = 4096$.

In the scale's case the reference voltages are power and ground, 0V and 3V, hence 3V has a quantization level of 4095 (4096-1) and 0V has a quantization level of 0. A general equation is given below for calculating the decimal value (quantization level) representing a voltage [16]:

$$N_{ADC} = (2^{\text{Re}\,solution} - 1) \times \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$$

V_{in} = Input voltage and V_{R+}, V_{R-} = Upper and lower reference voltages.

The General Purpose Input Output (GPIO) pins are digital input/output pins that can be used to determine whether a signal at a certain point in time is higher or lower than a set voltage threshold. If the voltage is higher than the threshold the GPIO register returns a logical ‘1’ or if lower it returns a logical ‘0’. Note that this assumes that the relevant GPIO register’s bit is set as input. In the example shown in Listing 7-1 the GPIO is set as output therefore setting a bit in the register as ‘1’ or ‘0’ determines the output of the associated GPIO pin (as high or low). There are 8 GPIO pins on the Wasa board used in this project. They are controlled by port 5. Each port has a number of control registers. The most interesting registers in our case are:

P5DIR	This selects whether a pin is output or input. A logical ‘1’ sets the pin as output.
P5SEL	This sets whether a pin is GPIO or an internal module.
P5IN	This holds the logical value at the input pin

7.2.2.1 *Sampling*

Sampling frequency is the rate at which the signal is captured, assuming that the signal is continuously sampled. The ADC allows different sampling modes. One of these modes is chosen by setting the second and first bits in the ADC control register ADC12CTL1as shown below.

ADC12CTL1 |= CONSEQx

The available modes are:

CONSEQ0	Single Channel sampled once (0x00)
CONSEQ_1	Sequence of Channels sampled once (0x01)
CONSEQ_2	Single Channel sampled continuously (0x02)
CONSEQ_3	Sequence of Channels sampled continuously (0x03)

8 Initial experiments

The experiments described in this section were conducted (2012-03-28) at the Department of Communication Systems (CoS) in the laboratory of Professors Smith and Maguire, in the Electrum building, located in Kista, Stockholm, Sweden. The lab provided all the tools and equipment used in the experiments.

8.1.1 Purpose

The purposes of these experiments were to gain some experience, to get an overview of the hardware used on the Wasa board, and to understand the potential uses of a device similar to the Wasa board. As we had very little experience in hardware, all of the experience from these experiments was of great help in planning what we would do for the rest of our thesis project. We especially wanted to see how the MSP430 worked and how it could be used. We also wanted to examine the use of sensors and to understand how this sensor data could be sent and displayed on a computer.

8.1.2 Approach

We started by investigating the Wasa board and its components. The components that were most interesting for us were the MSP430 microcontroller, FT232RL USB client controller, and the different sensors. We looked at how these components were connected and how additional sensors could be easily added to the board.

We read Prof. Mark T. Smith's description of the Wasa board version 1.7 [18] to gain an understanding of how we could read and display sensor data on our laptop computers. We downloaded the virtual COM port (VCP) driver suggested by Prof. Smith, installed a terminal to handle the message passing, and connected the Wasa board to the computer via a USB cable. It turned out that the VCP driver was actually unnecessary, as the Windows 7 operating system automatically downloaded the correct driver (v2.8.14.0) and installed it.

8.1.3 Communication

A computer can talk to the Wasa board via a USB interface. The board uses the FT232RL USB client controller and the drivers are available online for multiple platforms [19]. For our initial experiments we used a Windows 7 computer with the "Tera term terminal emulator" [20] installed. Prof. Mark T. Smith has an excellent guide [18] on his webpage that helped us with the setup.

We tested some of the commands that Prof. Mark T. Smith lists on his website. [21] It should be noted that all of these commands take the form of an "AT" command, such as one might use with a modem, but in this case these commands are interpreted by the software that was installed by Prof. Smith in the Wasa board.

8.1.3.1 *Accelerometer*

The accelerometer was tested by simply reading a value when the board was lying on the table and then reading another value when the board was held at an angle. This measurement measures the gravitational force as a vector, relative to the 3D accelerometer's axes. The command to get the accelerometer's current value is: AT+OAW? The command and its corresponding output for these two measurements were:

```
>AT+OAW?
+OAW: -2,-2,20
OK

>AT+OAW?
+OAW: -1,-21,-3
OK
```

We note the large change in the values of the Y and Z axes between these two measurements.

8.1.3.2 *Temperature sensor*

The temperature sensor was tested by first reading a control value and then rubbing a finger gently on the surface of the sensor to increase the sensor's temperature. The command to read the temperature sensor's value and its output for these two cases were:

```
>ATS204?
1946
OK

>ATS204?
1673
OK
```

Note that the resistance of the thermal sensor decreased, hence the voltage drop across the sensor (which determines the value read from this sensor) also decreased when the sensor was warmed. Given the specifics of this negative temperature coefficient sensor it is possible to translated the value to a temperature in degrees centigrade

8.1.3.3 *Light sensor*

We took several measurements with the light sensor on the Wasa board. First we took a control value, this value was given by just exposing the sensor to the normal light in the lab. We later covered the sensor with a finger and another value was output from sensor. These commands and their respective output values were:

```
>ATS203?
522
OK

>ATS203?
3853
OK
```

In this case the resistance of the light sensor increases when there was less light. If one use the specific characteristics of this sensor one could calibrate the sensor's output and get a light level in lux.

8.1.3.4 *Wasa Board info*

There is a command that returns information about the Wasa board. The "GMM" command returns the hardware version number and the "GMR" returns the software version number. These commands and their respective outputs were:

```
>AT+GMM?
+GMM: Wasa_1.7
OK

>AT+GMR?
+GMR: 12.25.2011
OK
```

8.1.4 **Code**

We took a quick look at the software written by Prof. Smith to gain some insight into how to program the MSP430. The software was written in C and included an MSP430x26x header file that

contains standardized identifiers and functions to program the MSP430. The code is available from Prof. Smith's homepage [22].

8.1.5 Results

These experiments gave us a basic understanding of how we could communicate with the Wasa board and the MSP430 to collect data. We also got a clearer picture of the components and the complexity of the Wasa board, which was useful to help us plan the next phase of our thesis project. This experiment also provides some basic knowledge about the tools we needed to use to communicate, program, and monitor the Wasa board and the MSP430. This experience also gave us greater confidence regarding our project and future work, which we needed.

8.1 Software experiments

We conducted several software experiments on 2012-03-30. These experiments are described in further detail in this section.

8.1.1 Purpose

Our goal was to investigate the software used in the Wasa board version 1.7. Prof. Smith provides both the source of the software [22] and a guide [23] about how to set up the programming environment and what tools are used to program the MSP430 microcontroller used on the Wasa board. Our main goal was to be able to load a modified version of the original software onto the board and to have this modified software operate as we expected.

8.1.2 Approach

By reading the guide that Prof. Smith provides on his website [23] and the "Users guide" that Texas Instruments provides on their website [24] we were able to modify the software to alter the output of a command. We decided to modify a simple string to see if we could successfully compile and install the modified software. The string we decided to change was the one that was displayed when reading the accelerometer. The function which outputs a string has the form: mts_puts(string). Thus we replaced *mts_puts("+OAW: ");*with*mts_puts("+OAW (TEST): ").*

As described by Prof. Smith, we had to download TI's Code Composer Essentials (CCE) to compile and generate a TI-TXT file to be loaded into the microcontroller. When searching for CCE at TI's homepage we found that they also referred to another program called Code Composer Studio (CCS). The CCS software is an updated and extended version of the CCE software, so we decided to download and use the CCS software instead. We modified the original software and made the proper settings (i.e., selecting the right microcontroller: MSP430F2618, and set the proper output format: TI-TXT format). We built the project to generate a TI-TXT output file. In order to load the modified code into the microcontroller we had to download a BSL scripter and write a script file to instruct the BSL scripter. We used the script proposed by Prof. Smith in his user guide [23] and only modified the name of the file that was to be loaded into the microcontroller. We ran the Biological Scripting Language (BSL) scripter together with the script file and (eventually) successfully loaded the modified program into the microcontroller.

8.1.3 Problems

We encountered several problems while conducting this experiment. The first problem that we ran into was a problem with the compiler version. We imported the original project written by Prof. Smith, which had been compiled by a different version of the CCS or CCE software than what we had on our computer. Because of this we could not build the project. We searched the Internet to find a solution to this problem, but found nothing. We eventually looked at the different options and settings of the CCS software and discovered the compiler version option and changed it from version 3.0.1 to version 4.1.0.B1, which solved the problem.

Another problem that we encountered was due to multiple connections to the Wasa board. Because we were already connected to the Wasa board by a terminal program, a new connection

could not be established. This became a problem when we tried to use the BSL scripter to load the modified software into the microcontroller. We looked at the errors that we got and thought it had something to do with the script file that we had written. We modified the script to try to solve the problem, but instead of solving the problem we introduced a new problem. Because we had modified the script to load the modified software into the microcontroller, this script aborted after a short period of time. To solve this problem we again stepped through all the instructions that were given by Prof. Smith and by TI and noticed that the script file was not correct. We modified the script, which solved the problem. The correct modified script is included as Appendix I.

8.1.4 **Results**

This experiment gave us a lot of new knowledge regarding the microcontroller and how we could program it. We learned how to build and load new software into the microcontroller by using CCS and BSL scripter. We also gained some experience about how new software for the microcontroller should be written.

The modified version of the microcontroller software ran as intended and displayed the expected modified output, as shown below:

Input:
>AT+OAW?

Result:
+OAW (TEST): 0,-2,22

9 Hello World

After gaining some understanding of the MSP430 and Wasa board I decided to write my own software. As a common first program I wrote a "Hello World" program. The aim was to continuously output "Hello World" to a terminal. The software was loaded into the microcontroller as explained in section 8.1. The first step was to setup the UART and Wasa board's initiation functions, these functions were available in the software provided by Prof. Smith (as mentioned in section 8.1). I wrote a write() function that is called by the main() function to copy the desired string to the UART's output buffer. After reading the MSP430 User Guide [17] this operation appeared to be quite easy. The data that is to be written is sent to a UART's output buffer (specifically UCA0TXBUF) that transfers it to the host that is connected to this serial port. The buffer processes one character at a time, thus to send a word or a sequence of data we have to loop over the characters in the buffer. To avoid overwriting the buffer's content a check is made to see whether the buffer empty flag is set *before* writing to the buffer. Listing 9-1 shows the pseudo code for this program.

```
Void write(char *string)
{
For (i=0;i< Size(string);i++)
{
While(!(Buffer.empty()));
UCA0TXBUF= string[i];
}
}
Main (){
Write("Hello World");
}
```

Listing 9-1: Hello World software

9.1 Results

Having used Prof. Smith's initiation functions this task turned out to be very simple. However, if I were to write my own initiation functions it would immediately be a lot more difficult as many things need to be considered, such as CPU clock rates, baud rates (symbols/second), selection of UART clock, and choice of and configuration of I/O pins.

As a result of writing this first program, I decided to use Prof. Smith's output functions in later programs as he has already implemented functions for outputting one character or a sequence of characters. His functions also handle the variable length of the string efficiently as he checks for the end-of-data '\0' rather than using an external C function. However, writing my own function made many things about writing programs for this processor much more understandable. The next step was a first attempt at writing a complete program to fetch the weight from the scale; this is described further in Chapter 10.

10 First program for reading data sent to the LCD

After familiarizing myself with the ADC, quantization levels, and the inner workings of the scale I made my first attempt at decoding the weight displayed by the LCD. My initial thought was to make use of the quantization levels and based on that calculating a corresponding digit. As discussed in section 6.3.2.1.2, if the difference between a backplane signal and a segment signal is above a voltage threshold then that segment is on. This was my starting point when writing my program. As a segment line and a backplane line intersect at only one segment the difference between them can be above a threshold only once per frame. If that segment is on; as shown in Figure 10-1, then my approach was to sample the lines 8 times (i.e., during 8 phases), calculate the total of the observed values in terms of decimal values (i.e., based on the quantization levels) and compute the difference between them. If the difference in values exceeds a given threshold, then that segment is on. In this case I wrote a function that returns a logical '1' otherwise a logical '0' based upon this calculation. The segment lines' values are compared for each backplane, thus 4 comparisons are done for each segment line. In total 8 bits are returned per digit (2 segment lines per digit) resulting in a value which is used to map to a corresponding digit. Listing 9-1 summarizes the pseudo code for the algorithm.

I needed to know what order the segments are with reference to the backplanes to be able to map the values to the corresponding digits. To find out the bit order I carried out a number of tests by connecting the pins, specifically one backplane and one segment to an oscilloscope and checked whether they were in opposition for a known digit at a given phase (see Figure 10-1). This same process was repeated for each of digit. Based on these measurements, I came to the conclusion that the segments are ordered as **degfxcba** (see Table 10-1). That is the bit order given when sampling at the beginning of a frame. By knowing this I could easily calculated what value is used to represent each digit. Table 10-1 summarizes the different values and digits.

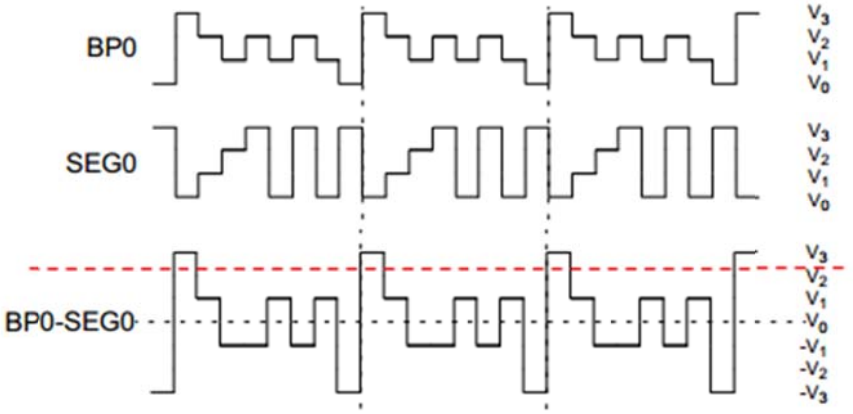

Figure 10-1: Difference between a segment- and a backplane line with a threshold of 2.3V

Table 10-1: Bit encoded digits

Digit	Bit encoding - degfxcba	Value
0	11010111	215
1	00000110	6
2	11100011	235
3	10100111	167
4	00111001	54
5	10110101	181
6	11110101	245
7	00000111	7
8	11110111	247
9	10110111	183

```
Signed int total (IO_line){
        Sample 8 times  (IO_line)
        Total = sample += sample+1;
        Return total;
}
Char segment_state(IO_1, IO_2){
    Difference(total(IO_1, IO_2)
    If(difference > threshold)
    Return 1;
    Else
    Return 0;
}
Void digit(){
Lcd_digit_0 = segment_state(backplane_1, segment_1) << 0
Lcd_digit_0 |= segment_state(backplane_2, segment_1) << 1
....
Lcd_digit_0 |= segment_state(backplane_1, segment_2) << 4
Lcd_digit_0 |= segment_state(backplane_2, segment_2) << 5
...

If(lcd_digit_0 = = 181)
Lcd_digit_0 = 5;
...
If(lcd_digit_0 = = 247)
Lcd_digit_0 = 8;

Weight = lcd_digit_0 x 0.1
Weight += lcd_digit_1 x 1
Weight += lcd_digit_2 x 10
Weight += lcd_digit_3 x 100

Return Weight
```

Listing 10-1: Weight decoding algorithm

10.1 Results

There were a few problems with this approach. The biggest problem being that there are 13 lines, but the microcontroller has only 8 ADC channels, of which only 6 are available on the Wasa board (as two are already used by the temperature and light sensor). This meant that all of the signals cannot be sampled this way. Prof. Maguire recommended taking a look at the GPIO pins and possibly using them as substitutes. However, as the algorithm depends heavily on the quantization levels this was not possible. Furthermore, sometimes the difference between two signals was negative and as the ADC

returns an unsigned integer it returned meaningless values. However, I kept on trying to solve these problems by converting the quantization values to bits by comparing them to yet another threshold to be able to use them with the GPIO pins. At that point it was time to implement timers and interrupts; as at this point the ADC sampled the signals immediately one after another. This caused further problems. Each time an interrupt was enabled the UART returned gibberish characters. It appears that the UART echos characters within the character stream causing problems. Having limited knowledge of interrupts I stopped and decided to make another attempt at decoding the weights by using the interrupt routines and AT commands written by Prof. Smith [22]. This second attempt is described in Chapter 11.

11 AT Commands

Before implementing my own AT command I needed a new strategy on how to decode the weights as I cannot rely solely on the ADC. After some further research on the LCD screen I came to the conclusion that I may not need the backplane signals at all and thus only need 9 pins, 8 of which could be GPIO pins (for the segment lines). The reason is that the backplane lines are there to tell whether a specific segment is on; however, this may not be needed if I were able to map different GPIO outputs to a range of known digits. The segments are only on when these segment lines are high within a certain duty cycle, thus I need only look at whether a segment is high at the sampled time. To be able to do so selecting the correct sampling rate is critical. The segment lines have to be sampled simultaneously at specific points in time. As explained in section 6.3.2.2 each segment line represents four segments. A segment line has 8 phases of which one duty cycle is 2 phases. A segment's state is determined during one of the four duty cycles (1 segment / duty cycle). That means that I only need to sample a segment line 4 times, during the second phase of a duty cycle. Having the segment lines connected to the GPIO pins the output would be a logical '1' and a logical '0' at the sampling time depending on a threshold. If it returns '1' that means that a segment is on at that time or '0' would mean that a segment is off at that time. Having 4 bits returned per pin and 2 pins per digit I would have a concatenated byte (8 bits). That byte would be used to map to different digits as done in the first attempt explained in Chapter 10. Figure 11-1 further clarifies the discussed strategy.

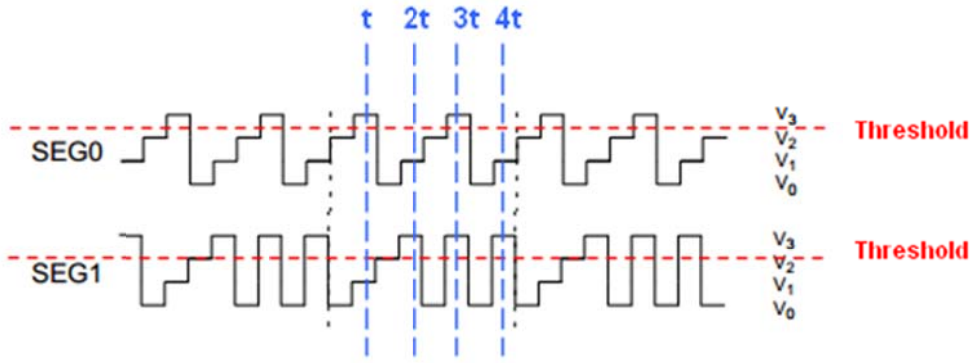

Figure 11-1: 4-sampling algorithm – least significant lines for the digit 2

Seg0 (xcba) : t = 1, 2t = 0, 3t = 1, 4t= 0

Seg1 (degf) : t = 0, 2t = 1, 3t = 1, 4t = 1

LSD (least significant digit): 01111010

Possible Bit encoding: fdegaxbc

Figure 10-1 shows the two least significant segment lines for the digit 2. The segments lines are sampled 4 times simultaneously at the specific points when the backplane lines would be in opposition. The specific points in time could be calculated by calculating the frame frequency and dividing it by 4, as that is how long each backplane line is connected to ground. The starting point to achieve the correct bit encoding is of less importance if all the segments are sampled simultaneously. For instance, as discussed in Chapter 10 I know that the actual bit encoding is **degfxcba,** then as shown in Table 10-1 the corresponding value for the digit '2' is 11100011, but in Figure 10-1 a possible bit encoding is **fdegaxbc**. The correct bit encoding can be determined when more digits are mapped. If all the segment lines were sampled simultaneously they would all follow the same bit pattern, thus the original bit encoding and backplanes are of less importance.

Based upon this strategy I implemented my own AT command to carry out this strategy. Section 11.1 gives a detailed description of this.

11.1 AT command implementation

I started by adding a function 'W' (W for weight) to the Wasa board's main switch-case function, that calls a getWeight() function which in turn returns a weight. The function getWeight() implements the algorithm discussed above. A segment line is sampled 4 times, once every duty cycle (once for every backplane). The segment lines are connected to the GPIO pins which means that each sample is either '0' or '1' corresponding to off or on. Each segment line is responsible for 4 segments, thus each sample corresponds to one segment and 8 samples (2 paired segment lines) correspond to a digit on the LCD display.

As discussed above the algorithm relies heavily on timing and accuracy. All the segment lines are sampled simultaneously which gives a certain bit order, which corresponds to a value and is mapped to a digit as shown in Table 10-1. However, as the sampling needs to be done specifically at the times when the segment lines are in opposition to the backplane lines I needed to know the frequency at which the lines were driven. To determine the frequency of the signals I connected a pin to an oscilloscope and determined the period it took the signal to complete one frame. The duration of a frame is ~20msec. That means that the frequency is 1/0.02 which is 50Hz, thus I decided to sample the signal at a frequency of 100Hz. Each backplane line is connected to ground for duration of 5msec, meaning that to sample at a 100Hz I need to sample every 2.5msec. Instead of using Timers and interrupts, I started by using the function '__delay_cycles()'. This function occupies the CPU doing nothing for a given number of CPU cycles. The Wasa board is configured to run at 16 MHz, thus one CPU cycle is 62.5µs and 2.5msec is 40000 cycles.

Having determined the sampling frequency I needed to define the sampling start time. The start time has to be consistent; otherwise the bit order would be different each time the AT command is executed. For future work the microcontroller should *not* wait for an AT command to get the weight, but rather run the function as soon as the scale is turned on. In that case the sampling start time could be fixed. However, when using AT commands it is impossible to determine beforehand when a user will execute the command, thus a fixed start time before sampling is not ideal. The sampling start time has to be triggered by some event. The first backplane signal is used to trigger the sampling. When the first backplane line is connected to ground the sampling should be begin. To determine when the first backplane line is connected to ground we continuously sample it using the ADC and when a sample is above a certain value the other sampling is executed, see Figure 11-2. As the backplane line is above the given threshold only once during a frame, the trigger occurs at the same time no matter when the AT command is executed.

Figure 11-2: Backplane sampling trigger. It triggers when the line is at its peak.

Due to overhead of the execution code is not ideal to start sampling immediately after the trigger. The reason is that I need to sample at the times when the segment line is in opposition to the backplane lines, as explained above. Therefore a fixed offset time is used after the trigger to start the sampling time at a point where the segment line is in opposition to the backplane lines. The segment line is then sampled four times every 2.5msec. See Figure 11-3 for further clarification.

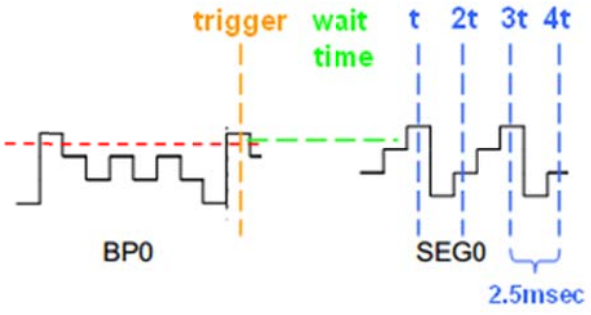

Figure 11-3: Sampling time

To determine the time offset I created a table with all the possible outcomes (a nibble of the four samples) from the third pin for the digit '0', as shown in Table 11-1 and Figure 11-4. If I were to sample immediately after the trigger the result from the third pin for the digit '0' is 0000. As seen in Table 11-1 there are four different start sampling points that returns that result. By adding an offset of 2.5msec the result was '1101', hence sampling immediately after the trigger the sampling starts at T_5, thus the overhead has duration of 15msec. I decided to have a time offset of 7.5msec to produce the bit order shown in Table 10-1, degfxcba.

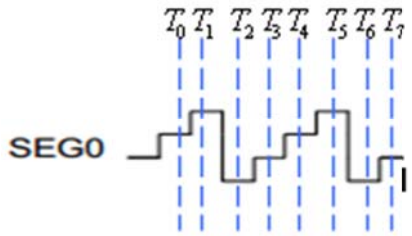

Figure 11-4: Possible sampling start times for the segment line for the third segment line for the digit '0'

Table 11-1: Possible outcomes for the third segment line for the digit '0'

T	Bit outcome
0	1011
1	0000
2	0111
3	0000
4	1110
5	0000
6	1101
7	0000

Having a desired bit order and sampling function I could easily map the different bytes (and corresponding values) to a digit, as shown in Table 10-1. This was done in the same way as the algorithm discussed in Chapter 10.

11.2 Measurement unit

The measurement unit (MU) was handled similarly to the weight; however the MU line was connected to an ADC pin. The MU line is driven in opposition to one backplane line depending on which unit is chosen, stones (st), kilograms (kg), or pounds (lb). If 'st' is chosen the MU line is in opposition to the first backplane line, 'kg' is in opposition to the second backplane line and 'lb' is in opposition to the third backplane line, as seen in Figure 11-5. Therefore the MU line will be high during only one duty cycle. What measurement unit is chosen can be calculated as I have done for the weight. Sampling the MU line is done in a similar way as sampling the segment lines. A trigger is triggered by the first backplane and an offset time of 7.5msec due to overhead is used. At that point I check the first duty cycle (t) which would be in opposition to the first backplane line if 'st' is chosen, by comparing it to a threshold, the same way I check for a trigger. If the first sample is not above the threshold then 'st' has not been chosen and I check the second duty cycle (2t) after a delay of 2.5msec, the same way the weights are calculated. If 'kg' is chosen the sample would be above the threshold and 'kg' is returned, otherwise the third duty cycle (3t) is sampled the same way and 'lb' is returned. The forth duty cycle holds no unit and is therefore ignored. Considering that a measurement unit has to chosen, one of the phases will always be high.

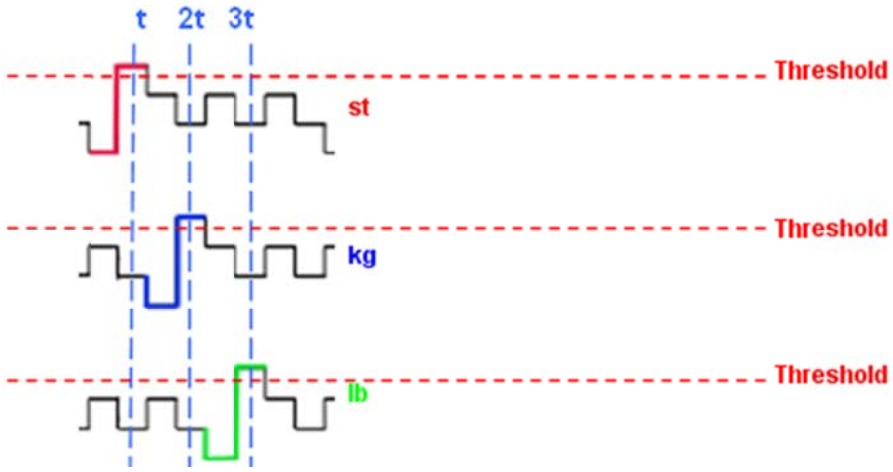

Figure 11-5: Measurement unit line. The line is HIGH during different duty cycles depending on which unit is chosen.

12 Send to local host

The code to derive the weight from the scale's LCD display is now complete. The next stage is to send the data to a receiver for further processing. Throughout this project I have discussed the different options to send the data wirelessly. However, before doing so I have decided to pass the weights via USB to an attached host. The reason was that it might not be possible to integrate a transceiver and send the data wirelessly within the given time frame of this thesis project, but it is still desirable to connect the sensor to a network. Therefore, I have decided to make use of the existing USB connection on the Wasa board to transfer the data to an attached computer for further processing.

Up until this point I had been using the terminal program 'Tera Term' to communicate with the Wasa board. However, I need to be able to receive the weights and do something useful with them, for instance saving the data locally to a file or perhaps entering it in a spreadsheet. The program could also be passed on to Andersson and Pedersen to send the data further to a server or a database.

The program that will run on the attached computer is written in C. The reason for choosing C is due to its great handling of hardware. The language provides many possibilities for easily communicating with serial ports and most importantly it provides portability to other operating systems. Nevertheless, writing the program in C makes it easier to integrate a similar program into the gateway built by López and Sánchez. The program uses the 'teuniz RS-232' library [25]. This is a highly recommended library for simple serial port communication. It is open source and multiplatform. It handles USB serial port emulating very well and is very easy to understand and use. Furthermore, it can be used with the gcc compiler tool chain on linux and MinGW on Windows without any changes. I wrote the program in Notepad++ and compiled it with MinGW. It is important to note that one must include the library's header file and source file in the compilation and make sure to disconnect the jumper from the 2 pin header to take the Wasa board out of BSL mode (if you previously used the BSL method to program the Wasa board). Another important note is that I have specifically wrote this program for Windows. To run this program on Linux you should change the include file 'Windows.h' to 'unistd.h' and the function 'Sleep()' should be replaced with 'usleep()'. This of course could be done by adding the appropriate conditional compilation directives to the file, but this is left to future work.

The program is quite simple. It starts out by settings up the serial communication. The code by default uses 8 data bits, no parity, and 1 stopbit; this can be changed in the library's header file ('rs232.h'). The program is set to communicate with COM4 (/dev/ttyS3 on linux) which is the port the Wasa board is connected to on my computer. The Wasa board is initially set to a baud rate of 115200 which is sufficient for my program. After setting the various settings the serial port is opened. A check is done to see whether opening the serial port was successful, otherwise an error message is returned. Upon successfully connecting to the Wasa board the program waits for the user to press the key 'w' (lower- case) to send the AT command 'atw' to the Wasa board, which calls the getWeight() function. It also creates or opens a text file to which the data will be written. An important note here is that I use the function getch() to get keyboard input. getch() is part of the conio.h library which is **not** a standard C library. This library is included in MinGW, but may cause some problems with gcc on linux or Cygwin on Windows. The recommended solution is to change getch() to getchar() or use ncurses [26]. The reason I did not initially use getchar() is that it requires the user to press 'ENTER' after each input, which I found undesirable. Receiving data from the serial port is done through polling. The program polls every 100msec until the number of characters in the input buffer exceeds zero. The received data is stored in a buffer, as well as written to the opened file. The received 'OK<CR><LF>' is stripped off the data (this string is returned at the end of every AT command). As a result only the weights are output, making it easy to parse the data for entry into a spreadsheet or another program. Andersson and Pedersen could send the data from the file or the buffer to a server with little modification to the program.

13 Market research

One of the main goals of this thesis project was to develop a scale with connectivity at a low cost. I have therefore decided to do research into the cost of the scale, the potentials for a connected scale, and comparisons to what is already available on the market today. The research is discussed in the following subsections.

13.1 Other scales available on the market

I searched the internet for different wireless scale solutions and was amazed at the wide range of devices that are already available. Most wireless scales that I found connect to a network through Bluetooth or Wi-Fi. One scale which I found to be interesting (as it is similar to my project) is the LifeSource UC-324THW [27]. The LifeSource scale records both weight and BMI and is connected wirelessly through a USB receiver that needs to be plugged to a computer. Their solution is very similar to the one I have considered. My scale is currently connected through wired USB, but if it had a wireless link it would depend on a unique gateway. In the case of the LifeSource scale, the USB receiver implements the gateway. Included in the purchase of the LifeSource scale is a 30-day trial for using their website based services for tracking and visual presentation of the data. Unfortunately, their website requires logging in to see their payment plans. I could not find a description of their payment plans elsewhere. This scale also requires a unique user, thus it is not multiuser friendly. Otherwise this scale is very similar in many ways to my solution and was the closest match I could find to my own solution. The LifeSource UC-324THW retails at €79 ($99.99) (as of 22 May 2012).

Other type of solution is the variety of Bluetooth connected scales. They ones I found to be the most interesting are the BC-590BT [28] and the iHealth HS3 [29]. The BC-590BT is the most expensive out of these two, retailing at €197.8 ($249.99) (as of 22 May 2012). The BC-590BT comes with a Bluetooth adapter allowing personal computer connectivity. Whereas the iHealth was developed solely for use with a device running Apple's iOS (i.e., a iPhone, iPod Touch, or iPad). In contrast, the BC-590BT could be connected to any Bluetooth capable device, including Android smartphones which provides the possibility to develop your own application. The BC-590BT uses Microsoft HealthValult to store its measurements. The iHealth comes with its own free application that can be downloaded from the AppStore, providing sharing possibilities as well. While both products are multiuser friendly, the iHealth is much less expensive and retails at €48.5 ($61.34) (as of 22 May 2012).

I personally believe that the best solutions are Wi-Fi connected scales. They are independent of customized gateways and devices and are always connected. They also provide the most reliable connection and are suitable for homes and larger facilities as Wi-Fi offers a much longer communication range. The Fitbit Aria [30] and the Withings [31] scales are the Wi-Fi scales which I found to be most interesting. The Fitbit Aria retails at €102.8 ($129.95) (as of 22 May 2012) and records weight, body fat, lean mass, and BMI and displays these values in neat graphs on their website or via a free iPhone, Android, or Windows phone application. The Fitbit Aria provides many different useful services. The user can set their goals, for example by making a calorie plan, as well as communicating these values to others. It is a very good looking scale. The applications provided are also very good looking. On the other hand the Withings scale has Android and iPhone support, as well a dedicated web service. The Withings scale records the weight, fat mass, lean mass, and BMI. It also provides many different features for generating different graphs and medical records showing height, growth, and other useful body measurements. The Withings scale integrates with Facebook and twitter allowing users to share their graphs with other people. It also offers a weekly email report. Both scales are multiuser friendly and support up to 8 users. The Withings scale retails at €129 (approx. $162). Table 13-1 summarizes the scales that have been discussed above in terms of communication range, features, multiuser friendliness, device support, and price.

Table 13-1: Scales summary

	LifeSource UC-324THW	BC-590BT	iHealth HS3	Fitbit Aria	Withings
Technology	Personal access point - USB	Bluetooth	Bluetooth	WIFI b/g	WIFI b/g
Range	15m	30m	30m	50m indoor	50m indoor
Included services	30 days trial web service.	Bluetooth adapter. Microsoft HealthVault	Free iOS application.	Free smartphones application. Wide range of website services.	Free smartphones application. Wide range of website services.
Multiple users	No	Yes – device dependent	Yes - device dependent	Yes – 8 users	Yes – 8 users
Device support	None	Bluetooth enabled devices	iPhone, iPod, Touch, iPad	iPhone, Android and Windows phone	iPhone, Android
Price	€79	€197.8	€48.5	€102.8	€129

13.2 The cost of this thesis project

The initial cost of the solution examined in this project is the Coline RTC3010 scale which has a retail price of €22 (199 SEK) at Chlas Ohlson (SKU: 34-5062). To this we have to add the cost of the sensor itself. Since I am using the Wasa board (version 1.7) I have to consider the cost of the additional features and sensors on the board. The resistors and non-polarized capacitors have cost the department of Communication Systems approximately €0.003 cents and the polarized capacitors cost on average €0.15. The light and temperature sensors on the Wasa board cost approximately €0.35 depending on manufacturer and volume. The accelerometer which is the most expensive sensor on the Wasa board costs approximately €1. The microcontroller (MSP430F2618) cost €10 and the USB controller €3. The PCB is a double sided copper blank board with a cost of approximately €0.02 cents/cm^2 and a total cost of approximately €1.30 (based upon a board size of 10 cm x 6.5 cm). Additionally there is the cost of the transceiver (TI CC1101) starting at €1.30 for quantity one. Crystals, connectors, and headers are an additional cost of roughly €1. That gives an approximate total cost of this project (including a transceiver) of approximately €40 which is still cheaper than the cheapest scale discussed above. However, this does not include production cost or application and service development and maintenance, nor does it include profit. However, many of the parts have been bought in low volume and the scale was purchased at retail price. Prof. Smith suggests that the USB controller will be included in the microcontroller in future builds of the Wasa board which would decrease the total cost of the project. The additional sensors are also unnecessary for this project. Removing the light, temperature, and accelerometer sensors would result in a saving of approximately €1.70. The Wasa board was developed for learning purposes and is therefore quite large. Prof. Smith suggests that the same board could be developed in the size of two 5 kronor coins. I believe that this could be reduced to the size of one 5 kronor coin if the additional sensors were removed.

Integrating a sensor into the scale during production would significantly decrease the cost of the scale (in comparison to adding it later). The additional cost for manufacturing is the parts of the sensor as well as the production cost of the sensor. Potentially the scale could be sold at €22 + sensor cost, a retail price of approximately less than €38. Profit (manufacture and Clas Ohlson), production cost, salaries, material cost, transportation, and other costs related to the process of selling the scale would be included in the original €22. Furthermore, as the scale now has a very useful feature the

value is increased and the price could slightly increase resulting in greater profit. However, there is also the cost of the gateway and application development and maintenance. I would expect that the gateway has a similar cost to the sensor of approximately €15.60 (with a PCB the size of one 5 kronor coin). Furthermore, I expect that the cost of developing and maintain a web service and/or a mobile application would be the salary of two developers (based on what Andersson and Pedersen are able to do during their parallel project) working full time during one year for every ten thousand units sold. I believe that setting a goal of selling ten thousand scales within a year nationwide is reasonable. An average programmer's salary in Sweden is €3,454 (31,101 SEK) per month (€41,448 per year) [32]. Including employee fees (at 31.42%) this would lead to a total cost of €108,942 for personnel, resulting in a cost of €10.9 per scale. A web host would cost on average €36 per year and €0.0036 per scale [33]. All in all the approximate total cost of a complete system with a scale, wireless transmitter, gateway, and a web service would be less than €64.5. This is €16 more expensive than the iHealth and €14.5 cheaper than the similar solution LifeSource UC-324THW. Table 13-2 summarizes the discussed costs.

Table 13-2: Total cost of developing a wireless scale

(a)

Part	Cost (€)
Cost of this project	
Coline RTC3010 scale	22 (199 SEK)
Resistors and non-polarized capacitors	0.003
Polarized capacitors	0.15
Light and temperature sensors	0.35
Accelerometer	1
Microcontroller (MSP430F2618)	10
USB controller	3
Double sided copper blank board (10 x 6.5 cm)	1.30
Transceiver (TI CC1101)	1.30
Crystals, connectors and headers	1
Total: 40.00 €	

(b)

Part	Cost (€)
Production cost of a wireless scale with an external sensor	
Coline RTC3010 scale	22 (199 SEK)
Resistors and non-polarized capacitors	0.003
Polarized capacitors	0.15
Microcontroller (MSP430F2618)	10
USB controller	3
Double sided copper blank board (the size of a 5 SEK coin – 6.47 cm^2)	0.13
Transceiver (TI CC1101)	1.30
Crystals, connectors and headers	1
Gateway	15.6
Web host	0.0036
Personal	10.9
Total: 64 €	

So far it is assumed that the sensor is externally built separated from the scale's PCB and microcontroller. In such case it appears that to have a complete solution with quality services and products it is difficult to keep the prices lower than what is already on the market. However, if the sensor would to be integrated into the scale's PCB and microcontroller the cost would significantly decrease. An algorithm for fetching the weight would be implemented into the scale's microcontroller, removing the need of the GPIO pins and connections to the LCD display, hence completely removing the need of an additional PCB. This would also eliminate the cost of all the other parts such as crystals, headers and USB controller. The only additional part to the scale would be the transceiver. Furthermore, the cost of the gateway, personnel and web host remains. The integration of the sensor into the scale's PCB and microcontroller would decrease the cost to €38.4, making this solution interesting and the scale a good competitor to the other scale already available on the market.

13.3 Market potentials

Network connected scales do have a significant potential market place, not only for personal bathroom scales but in the scientific and business sector as well. The many different scales work quite similarly and the same approach to fetch data is not much different. A major potential for network connected scales could be in retailing. A new legislation was introduced in Sweden 1 January 2010 that requires all retailers to obtain a certified cash register that is connected to a control unit [34]. The control unit reads all the registrations and outputs a control code that is controlled by the tax authorities. A network connected scale could significantly improve this process by transmitting the weight and cost of weight dependent items as well. Network connected scales could have a significant impact in the scientific and research sector as well. The different measurements could be transferred over a network or to a host for processing and visualization; this could significantly improve and simplify researchers' work. Floor scales are widely used in industries dealing with food products, measuring the amount of food produced or shipped. Network connected scale could for instance provide possibilities to control food production without being present in the factory. This could also be automated if the weights were uploaded to a control device that controls the amount of food. Network connected counting scales could be of great interest to businesses dealing without a cash register. The majority of amusement parks do not have cash registers in their staffed games, having to count the income by hand every night. For larger parks it would be highly interesting to be able to upload this data immediately and compute the daily income or other similar calculations. There is a big potential market for this type of scale and there are many opportunities in a wide range of fields that use scales.

14 Future work

There are many aspects of my work that can be improved or further developed. I have not been able to implement all of my ideas nor improvements due to time constraints, but may take it further in the future during my Master's thesis project. The timing can be improved by using timer modules and the power consumption can be significantly decreased. I have not been able to integrate a transceiver into my solution which would be highly desirable. This is certainly something that a future student may want to implement. The following subsections describe some suggested improvements and give further details of future potential developments that should be considered.

14.1 Wireless transmission

One of the main goals of this project was to make the sensor wireless, but due to time constraints this was not possible. The transceivers that were considered are the TI CC1101 and the TI CC1101-Q1 (low cost)[2]. The same transceiver is used in Sánchez's and López's gateway. The data could be transmitted in the 868 MHZ ISM-band using a proprietary protocol. A highly recommended protocol is SimpliciTI. The frame could include an identifier, the weight, and the measurement unit; as well as a hash value or CRC code for error detection. The data can be transmitted in many ways; however I would recommend bursting data every 10msec for duration of 1sec, hence bursting out the data 100 times. The reason is to minimize the risk of undelivered data. The modulation should also be Frequency shift keying (FSK). The receiver could catch as many of the transmissions as possible and save each frame in an array. The first frame (array element) should be checked using an error detection algorithm. If no errors were found, then the data will be extracted and sent to a server or database and the array emptied. If errors were found, then the receiver should check the second array element and so forth until one transmission is found that has no errors.

14.2 Timer modules

I have been using the function '__delay_cycles()' to control the timing. However, this may not be the ideal solution as it is dependent on the CPU clock, thus it is not portable to other systems without modifications, as well as being power consuming. The ideal solution is to use the 'Timer_A' module and go into low power mode during each delay. At the end of the time an interrupt will be called and the relevant code will be executed. Going into low power mode saves a lot of power. Figure 14-1 illustrates how much power can be saved by going into the different low power modes.

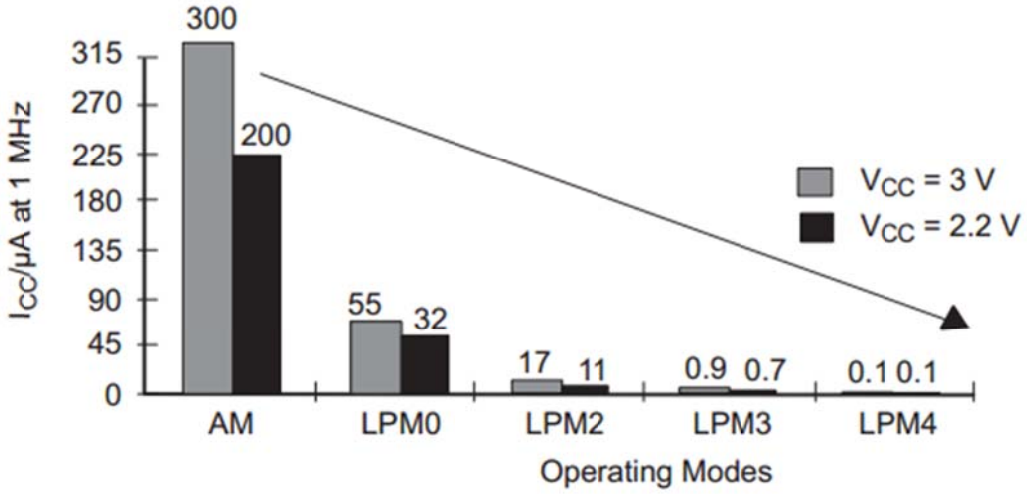

Figure 14-1: Low Power mode current consumption [17]

[2] Contact Prof. Maguire for more information, as some transceivers were bought for this project.

‘Timer_A’ uses the ACLK low-frequency clock which is enabled during low power mode 3 (LMP3), while all the other clocks are disabled; hence it is possible to significantly decrease power consumption by going from active mode to LPM3.

One time consuming mistake I made while attempting to use ‘Timer_A’ was to run in up mode. Up mode should only be used if you want continuous events independent of the execution of main, for instance blinking a LED every 1sec. It should not be used to programmatically trigger a delay. Another mistake I made was to never clear the CCIE (Capture/compare interrupt enable) bit inside the interrupt service routine causing periodic interrupts. TAR (Timer_A register) needs to be reset as well.

14.3 Power source

Studies into power consumption and power sources are important factors to consider in any future design. Most of the scales discussed in Chapter 13 uses 4 AA batteries, suggesting that wireless transmission and sensing are too power consuming for a CR2023 battery, hence the scale would need an additional power source. Piezoelectricity is quite interesting as a power source, but it might significantly increase the cost. Different power sources are discussed in Table 6-1.

14.4 Wi-Fi

The Wi-Fi solution is the most ideal wireless communication link. It offers high availability and a much longer range. It is also device independent and a very large number of homes already have Wi-Fi access points. A future development could be to integrate the gateway into the sensor, thus implementing an IP/TCP stack inside the sensor. Since the sensor would most likely use a transceiver the communication could occur in both direction which means that such a device could communicate via a Wi-Fi access point.

15 Results

The main focus of this thesis project has been to understand the analogue signals from the scale, in order to decode the weight measured by the scale. A sensor was developed that connects to the scale's LCD display and snoops the signals to decode the indicated weight. There are a total of 10 connections to the LCD display, 9 connections to the LCD's segment lines, one of which is responsible for the measurement unit, and 1 connection to a backplane line for timing purposes. The backplane and the measurement unit connections are connected to the microcontroller's ADC while the segment lines are connected to GPIO pins. The backplane line triggers a sampling of the GPIO pins after a set delay for calibration purposes. All the segments are sampled simultaneously 4 times at an interval of 2.5 msec. This solution can be used on any scale that has a multiplexed LCD display independent of the multiplexing ratio (number of backplanes) or bias ratio (number of voltage levels).

The decoded weight is sent via serial (USB) to a program that saves the data in a buffer and a text file. It is possible, without much modification of the code, to save the data in a spreadsheet in order to generate a graph locally. The program is portable to a different operating system with a small modification of the choice of header files. The program could easily be developed to send the data to a server or database for visualization on a website or a mobile phone. However, the system is limited to one unique user and can not distinguish between multiple users. Throughout this process I have documented tips, mistakes, and ideas to make it easier for future students to build upon my system.

16 Conclusions

I started out by looking into different sensors and hardware. A scale was the most interesting target and it seemed to be a reasonably easy device to work it. As I had very little hardware knowledge I started off looking into inappropriate solutions. I spent almost a week trying to decode the analogue signals as bit patterns, which turned out to be a huge misunderstanding. I did however learn many things throughout that week, such as using the different measurement apparatus available in the lab (specifically a multi meter and an oscilloscope) and also learned about the construction of a PCB. When I finally understood the inner working of the scale and its multiplexed LCD, I made my first attempt at decoding the lines. My main focus was the ADC, but as the ADC on the Wasa board only has 8 inputs, 2 of which were unavailable, it became quite difficult to find a solution using the ADC. Having looked at several other similar projects it appeared that the only possible solution was to use another microcontroller with a 16-channel ADC or use a multiplexer. When considering the GPIO pins, I found it difficult to understand how I could handle the mid-voltages which could return a logical value of '1' when I expected a '0'. I came to the conclusion that I needed to control the sampling with precise timing and skip the mid-voltage cycles. However, since I had no knowledge of embedded programming it turned out to be more difficult than anticipated to handle timers, clocks, and interrupts. In the end I decided to use a delay function and complete my solution before dealing with improvements that would use timers and interrupts. Unfortunately, I was never able to return to program the timers and interrupts due to time constraints.

One of my main goals was to enable wireless connectivity. However, decoding the scale took much longer time than I anticipated. Nevertheless, I still wanted to connect the scale to a network and thus decided to utilize a wired solution. I wrote a program in C that sends an AT command over the serial port that returns the weights at the time of the AT command execution. As of now the program saves the weights in a file and a buffer, but could easily be used to send the data over the internet to a server or a database.

Throughout this project I have tried and implemented many different ideas, many of which did not work. It appeared that working alone has its disadvantages as I had no one to consult with, meaning that I followed up on my ideas until the end. It was a good learning procedure to work until discovering my own mistakes, but it is very time consuming. I also believe that a better scale would have made the work much easier. A scale with an internal memory, allowing multiple users and a scale that records more than just a weight would have been a better scale to work it. Instead of dealing with snooping the LCD display I would fetch already formatted data directly from the scale's memory. By doing so I would most likely have had time to focus on the transceiver, rather than the scale itself. On the other hand, the path that I took proved to be a useful learning experience.

This thesis project has been very educational and has allowed me to explore a new field in computer science. I believe that the newly acquired knowledge will prove itself to be very useful for my Master's degree in the field of communication systems.

16.1 Required reflections

There are many aspects to be considered when developing a network connected scale. The society and a user could be impacted in many ways. The economics of businesses could be improved with higher accuracy and decreased wastage. A network connected scale could help in gathering statistics and be a great tool for researchers and schools. Wastage, especially in the food industry could be significantly reduced using network connected scales. Schools could also decrease food wastage using this type of scales and the services with it by monitoring and adapting to the amount of food being eaten and wasted. This would improve upon the economics of the schools and society as a whole. It would also have a significant impact on the environment with less wastage. These types of scales could be of use to the tax authorities preventing fraud and cheating, as well as simplifying the control of wider range of items.

As weights could be sensitive information the data should be encrypted already at the transmitter. Since the transmitter would most likely use the 868 MHz band it is possible for anyone to sniff the data without much difficulty, this further emphasizes the need of encryption. The weights, especially for a personal bathroom scale, should be respected and only shared with the user's consent. Any visualization of the weights should be within the user's control and any personal information should not be sold or shared to third parties.

References

[1] Ericsson, 'More than 50 billion connected devices'. Available at http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf, [accessed May 6, 2012].

[2] Prof. Gerald Q. Maguire, 'Examples of thesis projects'. [Online]. Available: http://web.it.kth.se/~maguire/maguire-exjobbs-examples.html. [Accessed: 18-April-2012].

[3] Jorge Pinto, 'Digital bathroom scale which logs weight and data on SD Card'. [Online]. Available: http://code.google.com/p/casainho-projects/wiki/SdCardBathroomScale#Technical_details_--_Scale_Jata_Hogar_490. [Accessed: 06-May-2012].

[4] Micah Elizabeth Scott, 'Hacking a Digital Bathroom Scale'. [Online]. Available: http://scanlime.org/2010/01/hacking-a-digital-bathroom-scale/. [Accessed: 06-May-2012].

[5] Jorge Pinto, 'A Open Source bathroom scale with Bluetooth communication to Android Smart Phones.' [Online]. Available: http://code.google.com/p/casainho-projects/wiki/SmartScale. [Accessed: 06-May-2012].

[6] Albert López and Francisco Javier Sánchez, 'Exploiting Wireless Sensors', Master thesis, KTH, Royal Institute of Technology, School of Information and Communication Technology, Stockholm, Sweden, Not yet published.

[7] A. Andersson and I. Bremstedt Pedersen, 'More than downloading', Bachelor's thesis, KTH Royal Institute of Technology, June 2012.

[8] T. Petruzzellis, *Electronic Sensors for the Evil Genius*. McGraw-Hill, 2006, ISBN: 9780071470360.

[9] W. Stallings and W. Stallings, *Wireless communications and networks*. Upper Saddle River, NJ: Pearson Prentice Hall, 2005, ISBN: 0131918354 9780131918351 0131967908 9780131967908.

[10] Texas Instruments, 'SimpliciTI'. [Online]. Available: http://www.ti.com/tool/simpliciti. [Accessed: 03-May-2012].

[11] Quicksupply, 'How Does A Digital Scale Work?' [Online]. Available: http://www.quicksupply.net/t-how-to-scale.asp. [Accessed: 03-May-2012].

[12] M. Trimarchi, 'Piezoelectricity'. [Online]. Available: http://science.howstuffworks.com/environmental/green-science/house-music-energy-crisis1.htm. [Accessed: 09-April-2012].

[13] Rodger Richey, 'LCD Fundamentals Using PIC16C92X Microcontrollers'. Available at http://ww1.microchip.com/downloads/en/AppNotes/00658a.pdf, [accessed April 16, 2012].

[14] Zilog, 'The Z8 Encore!® MCU as an LCD Driver'. Available at http://www.zilog.com/docs/z8encore/appnotes/an0162.pdf, [accessed April 16, 2012].

[15] T. Hådén, *IPv6 Home Automation*. 2009, Available at http://kth.diva-portal.org/smash/record.jsf?pid=diva2:510500.

[16] Gustavo Litovsky, 'Beginning Microcontrollers with the MSP430 Tutorial'. Available at http://www.glitovsky.com/Tutorialv0_3.pdf, [accessed April 9, 2012].

[17] Texas Instruments, 'MSP430x2xx Family - User's Guide'. Available at http://www.ti.com/lit/ug/slau144i/slau144i.pdf, [accessed April 9, 2012].

[18] Prof. Mark T. Smith, 'Wasa Board version 1.7 Description', *Wasa 1.7 Desciption*. [Online]. Available: http://web.it.kth.se/~msmith/wasa/wasa1.7/Wasa_1-7_Description.html.

[19] Future Technology Devices International, *Viritual COM port (VCP) driver*. Available at http://www.ftdichip.com/Drivers/VCP.htm.

[20] Yutaka Hirata, Iwamoto Kouichi, and Takashi Teranishi, *TeraTerm Project*. Available at http://ttssh2.sourceforge.jp/, [accessed April 18, 2012].

[21] Prof. Mark T. Smith, 'Wasa 1.7 AT Command Details'. Available at http://web.it.kth.se/~msmith/wasa/wasa1.7/wasa_1-7_AT_command_map.pdf, [accessed April 18, 2012].

[22] Prof. Mark T. Smith, *Wasa Board v1.7 at cmmds*. 2012, Available at http://web.it.kth.se/~msmith/wasa/wasa1.7/WasaBoardv1.7 atcmmds.zip.

[23] Prof. Mark T. Smith, ‘How to Load Programs onto the Wasa Board’. Available at http://web.it.kth.se/~msmith/wasa/wasa1.7/how_to_load_programs_wasa_board.pdf, [accessed April 18, 2012].
[24] Texas Instruments Incorporated, ‘Code Composer Studio™ v4.2 User’s Guide for MSP430™’. August-2005.
[25] Teunis van Beelen, ‘RS-232 for Linux and Windows’. [Online]. Available: http://www.teuniz.net/RS-232/. [Accessed: 24-May-2012].
[26] Godhc, ‘Include Conio.h In Linux – Traditional C Programs’. [Online]. Available: http://godhc.wordpress.com/2011/01/16/include-conio-h-linux-ubuntu/. [Accessed: 24-May-2012].
[27] Amazon, ‘LifeSource UC-324THW Wireless Scale’. [Online]. Available: http://www.amazon.com/LifeSource-UC-324THW-Wireless-Scale/dp/B001IDY5IQ. [Accessed: 24-May-2012].
[28] Tanita Corporation, ‘Tanita Corporation - BC590BT’. [Online]. Available: http://www.thecompetitiveedge.com/shop/item/123-productId.184549627_123-catId.176160849.html. [Accessed: 24-May-2012].
[29] Amazon, ‘iHealth HS3 Wireless Bluetooth Scale’. [Online]. Available: http://www.amazon.com/gp/product/B005FPT51U/ref=pd_lpo_k2_dp_sr_2?pf_rd_p=1278548962&pf_rd_s=lpo-top-stripe-1&pf_rd_t=201&pf_rd_i=B001IDY5IQ&pf_rd_m=ATVPDKIKX0DER&pf_rd_r=13HR9D1MPPR35WMFZF5H. [Accessed: 24-May-2012].
[30] Fitbit, ‘Fitbit Aria Wi-Fi Smart Scale’. [Online]. Available: http://www.fitbit.com/product/aria. [Accessed: 24-May-2012].
[31] Withings, ‘Withings scale’. [Online]. Available: http://www.withings.com/en/bodyscale/features. [Accessed: 24-May-2012].
[32] Lönestatistik, ‘Programmerare löner’. [Online]. Available: http://www.lonestatistik.se/loner.asp/yrke/Programmerare-1001. [Accessed: 24-May-2012].
[33] findmyhosting, ‘Web Hosting Side By Side Comparison’. [Online]. Available: http://findmyhosting.com/compare-hosting/. [Accessed: 10-June-2012].
[34] Skatteverket, ‘Kassaregister | Skatteverket’. [Online]. Available: http://www.skatteverket.se/foretagorganisationer/startadrivaavslutaforetag/kassaregister.4.121b82f011a74172e5880005263.html. [Accessed: 10-June-2012].

Appendix I

The modified download script

```
case'W':
if(action != '?') {
      ret_value = 0;
      break;
           }
do {
select_and_read_I2C_B0_256( 0, accel_data, 3 );
          } while ( (accel_data[0] & 0x40) || (accel_data[1] & 0x40) ||
(accel_data[2] & 0x40) );
if (accel_data[0] & 0x20) accel_data[0] |= 0xc0;
if (accel_data[1] & 0x20) accel_data[1] |= 0xc0;
if (accel_data[2] & 0x20) accel_data[2] |= 0xc0;
mts_byte2str((signedchar) accel_data[0], working_buffer);
if( basic_at_command_flags&flag_v ) mts_puts("\r\n");

mts_puts("+OAW: (TEST)");// changed from mts_puts("+OAW: ");

mts_puts(working_buffer);
mts_byte2str((signedchar) accel_data[1], working_buffer);
mts_puts(",");
mts_puts(working_buffer);
mts_byte2str((signedchar) accel_data[2], working_buffer);
mts_puts(",");
mts_puts(working_buffer);
mts_puts("\r\n");
break;
```

Appendix II

Hello World

```
void write(char * str)
{
int i = 0;
for(i = 0; i < strlen(str); i++)
{
while (!(IFG2&UCA0TXIFG)); // USART0 TX buffer ready?
UCA0TXBUF = str[i];
}
}
volatile unsigned int i;
void main(void) {

            WDTCTL = WDTPW+WDTHOLD;
            /* Found in the code provided by Prof. Smith*/
            init_wasa_board();
            set_up_UART_A0();
            for (;;)
            {
            i = 500000; // Delay
            do (i--);
            while (i != 0);
            write("Hello World\n");
            }
}
```

Appendix III

First weight decoding software

```
#define   Num_of_Results   8

unsigned int i;
unsigned int results[Num_of_Results];

unsigned int segv_total(int io_number){
      unsigned int seg_total = 0;
      switch (io_number)
      {
      case 0:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM0;              // Move results
                  seg_total += results[i];
            }
            break;
      case 1:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM1;              // Move results
                  seg_total += results[i];
            }
            break;
      case 2:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM2;              // Move results
                  seg_total += results[i];
            }
            break;
      case 3:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM3;              // Move results
                  seg_total += results[i];
            }
            break;
      case 4:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM4;              // Move results
                  seg_total += results[i];
            }
            break;
      case 5:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM5;              // Move results
                  seg_total += results[i];
            }
            break;
      case 6:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM6;              // Move results
                  seg_total += results[i];
            }
            break;
      case 7:
            for(i =0; i<8;i++){
                  results[i] = ADC12MEM7;              // Move results
                  seg_total += results[i];
```

```
            }
            break;
      }

      return seg_total;
}
unsigned int segv_difference(int io_number, int io2_number){

      return segv_total(io_number) - segv_total(io2_number);
}

unsigned char seg_bit(int io_number,int io2_number)
{
      if (segv_difference(io_number,io2_number) > 1884)
            return 1;
      else
            return 0;
}
char number_to_digit (unsigned char *number, unsigned char *digit)
{
    switch (*number)
    {
        case 0:
        *digit = 0;
        break;

        case 215:
        *digit = 0;
        break;

        case 6:
        *digit = 1;
        break;

        case 235:
        *digit = 2;
        break;

        case 167:
        *digit = 3;
        break;

        case 54:
        *digit = 4;
        break;

        case 181:
        *digit = 5;
        break;

        case 245:
        *digit = 6;
        break;

        case 7:
        *digit = 7;
        break;

        case 247:
        *digit = 8;
        break;

        case 183:
```

```
        *digit = 9;
        break;

        default:
        return 1;
    }

    return 0;
}


void digit ()
{
      unsigned char   lcd_input_digit_0;
      unsigned char     digit;
      float *weight;
    lcd_input_digit_0 = ((seg_bit(9,0)) << 0);
      lcd_input_digit_0 |= ((seg_bit(10,0)) << 1);
      lcd_input_digit_0 |= ((seg_bit(11,0)) << 2);
    lcd_input_digit_0 |= ((seg_bit(12,0)) << 3);
    lcd_input_digit_0 |= ((seg_bit(9,1)) << 4);
    lcd_input_digit_0 |= ((seg_bit(10,1)) << 5);
    lcd_input_digit_0 |= ((seg_bit(11,1)) << 6);
    lcd_input_digit_0 |= ((seg_bit(12,1)) << 7);

    if (number_to_digit (&lcd_input_digit_0, &digit))
    {
        *weight = weight_bck;
        return 1;
    }

    *weight += digit * 0.1;
    //Write function found in the code provided by Prof. Smith.
    mts_puts(weight);
}

void main(void)
{
      WDTCTL = WDTPW+WDTHOLD;                   // Stop watchdog timer
      init_wasa_board();
      set_up_UART_A0();
        P6DIR = 0x00;             // Be sure all are inputs
        P6SEL = 0xff;            // and all are ADC channel inputs
      ADC12CTL0 = ADC12ON+SHT0_8+MSC;           // Turn on ADC12, set
sampling time
      ADC12CTL1 = SHP+CONSEQ_3;                 // Use sampling timer, set
mode
      ADC12IE = 0x01;                           // Enable ADC12IFG.0
      ADC12CTL0 |= ENC;                         // Enable conversions
      ADC12CTL0 |= ADC12SC;                     // Start conversion

      digit ();

}
```

Appendix IV

Second weight decoding software – AT command

```
            volatile int i;
            volatile unsigned short Samples[10];


void timer_a(){
                  __delay_cycles(80000);
            }

            unsigned char pin_state(int i, int GPIO_pin_number){
                  char bit_value;
                  bit_value = ((Samples[i] & (1 << GPIO_pin_number)) ?
1:0);

                  return bit_value;
            }

            char number_to_digit (unsigned char *number,unsigned char
*digit)
            {
                  switch (*number)
                  {

                  case 215:
                        *digit = 0;
                        break;

                  case 6:
                        *digit = 1;
                        break;

                  case 235:
                        *digit = 2;
                        break;

                  case 167:
                        *digit = 3;
                        break;

                  case 54:
                        *digit = 4;
                        break;

                  case 181:
                        *digit = 5;
                        break;

                  case 245:
                        *digit = 6;
                        break;

                  case 7:
                        *digit = 7;
                        break;

                  case 247:
                        *digit = 8;
```

```
                        break;

                  case 183:
                        *digit = 9;
                        break;

                  default:

                        return 1;
                  }

                  return 0;
            }
int getWeight(void){
                  P6DIR = 0x00;              // Be sure all are inputs
                  P6SEL |= 0x01;            // and all are ADC channel
inputs
                  ADC12CTL0 = SHT0_2 + ADC12ON+MSC; // Sampling time, turn on
ADC12
                  ADC12CTL1 = SHP + CONSEQ_2;
                  ADC12CTL0 |= ENC;                     // Enable
conversions
                  ADC12CTL0 |= ADC12SC;                 // Start conversion

                  P5SEL = 0;
                  P5DIR &= ~0xFF;
                  char working_buffer[10];

                  while(1){
                        if(ADC12MEM0 >  3200){
                              __delay_cycles(120000);
                              for (i = 0; i < 4; i++) {
                                    Samples[i]=P5IN;
                                    timer_a();
                              }
                              break;
                        }
                        __delay_cycles(40000);
                  }

                  unsigned char   lcd_digit_0, lcd_digit_1, lcd_digit_2,
                  lcd_digit_3;
                  unsigned char     digit;
                  float weight = 0.0;

                  lcd_digit_0 = ((pin_state(0,1)) << 0);
                  lcd_digit_0 |= ((pin_state(1,1)) << 1);
                  lcd_digit_0 |= ((pin_state(2,1)) << 2);
                  lcd_digit_0 |= ((pin_state(3,1)) << 3);
                  lcd_digit_0 |= ((pin_state(0,3)) << 4);
                  lcd_digit_0 |= ((pin_state(1,3)) << 5);
                  lcd_digit_0 |= ((pin_state(2,3)) << 6);
                  lcd_digit_0 |= ((pin_state(3,3)) << 7);

                  lcd_digit_1 = ((pin_state(0,1)) << 0);
                  lcd_digit_1 |= ((pin_state(1,1)) << 1);
                  lcd_digit_1 |= ((pin_state(2,1)) << 2);
                  lcd_digit_1 |= ((pin_state(3,1)) << 3);
                  lcd_digit_1 |= ((pin_state(0,3)) << 4);
                  lcd_digit_1 |= ((pin_state(1,3)) << 5);
                  lcd_digit_1 |= ((pin_state(2,3)) << 6);
                  lcd_digit_1 |= ((pin_state(3,3)) << 7);
```

```
                lcd_digit_2 = ((pin_state(0,1)) << 0);
                lcd_digit_2 |= ((pin_state(1,1)) << 1);
                lcd_digit_2 |= ((pin_state(2,1)) << 2);
                lcd_digit_2 |= ((pin_state(3,1)) << 3);
                lcd_digit_2 |= ((pin_state(0,3)) << 4);
                lcd_digit_2 |= ((pin_state(1,3)) << 5);
                lcd_digit_2 |= ((pin_state(2,3)) << 6);
                lcd_digit_2 |= ((pin_state(3,3)) << 7);

                lcd_digit_3 = ((pin_state(0,1)) << 0);
                lcd_digit_3 |= ((pin_state(1,1)) << 1);
                lcd_digit_3 |= ((pin_state(2,1)) << 2);
                lcd_digit_3 |= ((pin_state(3,1)) << 3);
                lcd_digit_3 |= ((pin_state(0,3)) << 4);
                lcd_digit_3 |= ((pin_state(1,3)) << 5);
                lcd_digit_3 |= ((pin_state(2,3)) << 6);
                lcd_digit_3 |= ((pin_state(3,3)) << 7);

                if (!(number_to_digit (&lcd_digit_0, &digit)))
                {
                      weight += digit * 0.1;
                }
                if (!(number_to_digit (&lcd_digit_1, &digit)))
                {
                      weight += digit * 1;
                }
                if (!(number_to_digit (&lcd_digit_2, &digit)))
                {
                      weight += digit * 10;
                }
                if (!(number_to_digit (&lcd_digit_3, &digit)))
                {
                      weight += digit * 100;
                }
                snprintf(working_buffer, 9, "%3.1f", weight);
                mts_puts(working_buffer);
                measurement_unit(working_buffer);
                mts_puts(working_buffer);

                return 1;

          }
```

’W’ function in the main switch-case statement

```
case 'W':
      cmd_ok = getWeight();
      break;
```

Appendix V

Measurement unit source code

```
char measurement_unit(char *unit){

      while(1){
            if(ADC12MEM0 >  3200){
                  __delay_cycles(120000);
                  if(ADC12MEM1 > 3200){
                        unit = "st";
                        break;
                  }
                  timer_a();
                  if(ADC12MEM1 > 3200){
                        unit = "kg";
                        break;
                  }
                  timer_a();
                  if(ADC12MEM1 > 3200)
                        unit = "lb";
                  break;
            }
            __delay_cycles(40000);
      }
      return 0;
}
```

Appendix VI

Local host program

```
#include <stdlib.h>
#include <stdio.h>
#include <Windows.h>
#include <conio.h>
#include "rs232.h"

int main()
{
  int i, n,
        port_nr=3,        /* COM4 on windows */
        bdrate=115200;       /* 115200 baud */

  unsigned char working_buffer[512];
unsigned char path[] = "e:\\kexjobb\\scale_c\\weights.txt";
FILE *file_pointer;

  if(OpenComport(port_nr, bdrate))
  {
      printf("Unable to open comport\n");
      return(0);
  }
  //wait for keypress 'w' before opening file and send AT command

  char ch = getch();
  while (ch != 'w'){
  ch=getch();
  }

  file_pointer=fopen(path, "a"); // open file
      cprintf(port_nr, "atw\r\0"); // send AT command to serial port

  while(1)
  {
      n = PollComport(port_nr, working_buffer, 511); //read data by polling
every 100msec
      if(n > 0)
      {
        working_buffer[n] = 0;   /* null-terminated */
             strtok(working_buffer, "OK\r\n");
        fprintf(file_pointer, "%s\n",(char *)working_buffer); /* write to
file */
        printf("Weight saved!");
        fclose(file_pointer);
        CloseComport(port_nr);
        break;
      }
      Sleep(100);
  }

  return(0);
   }
```

TRITA-ICT-EX-2012:76

www.kth.se