Wireless Sensor Networks

TU Delft

Faculty of Science of Education

E.H. van Tol-Homan

7st of March, 2013

Abstract

Wireless sensing networks (WSN) are becoming more and more popular in different domains: e.g. sport players would like to measure the results of their training immediately. Physiotherapist would immediately like to measure the results of the treatment of their client, during recovering. Industrial companies would like to follow the production process.

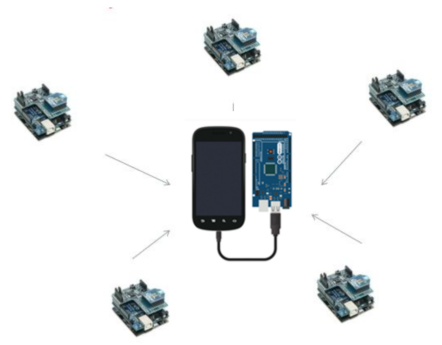
This work has answered the following research questions:

How to develop a low-cost, reusable, and easy-to-use wireless sensing network for a given use-case?

A prototype has been build, with open-source hardware and software, which demonstrates the possibility of using the Arduino platform for WSN.

What is the best architecture to meet the requirements?

Research is done on different hardware architectures. As a result of this, the following architecture structure is used in the prototype, a combination of Xbee, USB and 3G.



How to design a common protocol for sending sensor data?

The standardization is realized using a streaming protocol for sending and receiving data.

A prototype is developed for the Dispena football application FootballNote.com, where football players can measure their exercises on the football field. The developed system will be used as a base for further development for Dispena, as well be integrated into their software solutions.

Index

Table and figures

1 Introduction

With the upcoming technique of wireless sensing networks (WSN) it becomes possible to seamlessly couple the physical environment with the digital world. Since it became possible to develop low-cost wireless networks, the implementation of wireless sensor networks to monitor physical or environmental conditions has become possible, such as temperature, sound, pressure etc., and to cooperatively move data through the network to a main server.

1.1 The essentials of wireless sensing networks (WSN)

Wireless sensor networks make it possible to integrate devices with the virtual world of the Internet, and to interact by tracking, sensing, and monitoring objects and their environment. The essence of wireless sensor networks is that a wireless sensor networks device can generate and retrieve information about and from objects and their environment. Wireless sensor networks combine three pieces of equipment: a microcomputer, a sensor, and wireless data communication. A sensor is a device that can measure a physical phenomenon (like temperature, radiation, humidity). A list of wireless sensing networks elements relevant for low-cost development is given here.

In the 1960's, the standard cable protocol RS-232 was created and published by the Electronics Industry Alliance (EIA). It is a standard interface for data streaming.

In 1994, Bluetooth started a wireless technology standard for exchanging data over short distances, using short wave-length radio transmission for fixed and mobile devices. The created network was called "Personal Area Network". The main difference with the standard cable protocol RS-232 is that Bluetooth can connect several devices simultaneously.

In 1999, another standard had been developed, i.e. Wi-Fi. This technology allows an electronic device to exchange data using radio waves over a computer network, and is based on the 802.11 standard.

In 2005, a company started to develop the Xbee which was based on the 802.15.4-2003 standard. This standard is designed for point-to-point and point-to-multipoint communication. Between 2005 and 2011, various versions of Xbees were developed using the Zigbee protocol, using a "Personal Area Network" similar to that of Bluetooth.

Also in 2005, in Ivrea (Italy), a prototype platform –called Arduino- was initiated (at the Olivetti company). With the Arduino it became, possible to make a cheap prototype of a system for electronics system development. The founders of Arduino - developers at Olivetti - Massimo Banzi and David Cuartielles named the prototype "Arduino" after Arduin of Ivrea, which also means "brave friend". Information about the Arduino can be found in Appendix A.

In May 2011, a total of 300,000 Arduino's had been sold; more and more people started using an Arduino for measuring an object's environment. In combination with the above mentioned techniques (Bluetooth, Wi-Fi and Xbee) projects were initiated to make wireless sensing networks.

Combining the Arduino and wireless communication, low cost wireless sensor networks became possible. In 2011, companies started to provide services such as Pachuba to store data. Pachuba is a web service that enables someone to store, share real-time sensor data, energetic and environmental data from objects, devices and buildings around the world.

Once a successful prototype had been built with the above techniques, it had to be translated into an electronic scheme in order to get it into production. In order to build the electronic scheme, the graphical editor Fritzing can be used. Fritzing is an open source program designed for bringing a prototype into production. It generates schematic PCB (Printed Circuit Board) production files, which can then be produced.

No matter what device is used, the idea behind wireless sensor networks is to measure the physical world around the object. The goal is to seamlessly couple the physical environment with the digital world, thus enabling a user to gather information about physical phenomena.

1.2 Problem description

Dispena Solutions[1] would like to develop applications for wireless sensor networking and to provide private data storage. An internal report from Dispena Solutions gives an overview of possible wireless sensor networking systems (Appendix B), and these use-cases form the basis of this research.

There are two main branches, where this company would like to use WSN:

- Sport: Measuring individual sport performance.

  When the sport performance of individual players is measured, the training can adapted, based on the test results. Dispena developed a training program for football (soccer) players. Embedded in this application, Dispena would like to measure the sport performance of the players. In figure 1, an example is given of what has to be measured during a slalom. Dispena would like to have a system, which measures the intermediate time of a slalom continued by measuring the shooting power, when shooting the football into the goal.

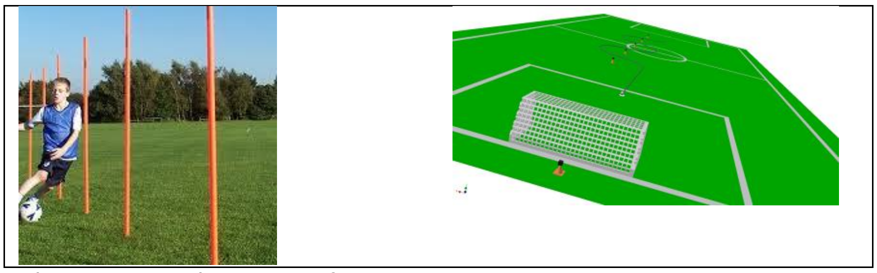


Figure 1: Measuring sport performance

  Short list of activities: Measure time at slalom at pole 0 to 5; Measure speed of ball; Display Result of Single Player; Send values into the database.

- Industry: Measurements during production processes.

  Dispena would like to have a low-cost system for measuring a part of the production process. After an object is identified by reading a RFID-chip, a list of physical phenomena must be measured. In figure 2, an object is identified followed by measuring e.g. temperature, weight, or size.

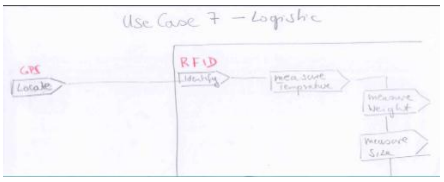


Figure 2: Measuring sport performance

[1] The company where this research is initiated.

The two given use-cases are totally different, but the way of measuring has the following in common:

- Each sensor is physically in another place, within 40 meters.
- The data are read in sequence. In case of the sport example, the times when reaching pole one, two, three, four and five are measured in a sequence. In the second example, after identification the temperature and weight are measured.
- The use-case use the following sensor types:
    - Sensors, which detect the occurrence of a phenomenon, such as movement, in values of 1 or 0 (movement vs. no movement) at a certain point in time. These sensors are called: Digital Sensors. Digital sensors produce a binary output signal in the form of a logic “0” or “1” per bit. The bits can be combined to make a byte (parallel transmission).

      A common example is a switch: which generates one of two states: on / off; or in the football example, detecting that the player has passed a pole.
    - Sensors, which measure the strength of the physical phenomenon, where the timing is less important. These are called: Analog Sensors. An analog sensor, such as a thermistor (a resistor where the resistance depends on the temperature), is integrated into a circuit, which will output a specific voltage, usually between 0 and 5 Volts. The signal is continuous in time and amplitude. The output is converted with an Analog-Digital Converter into a numeric value. These type of sensors could measure for example: Acceleration (Measuring the acceleration in x, y and/or z.), temperature, humidity, distance, light, weight, etc.

1.3 Key requirements

Together with Dispena Solutions, a list is made with the requirements (table 1) that should be met for their use-cases.

Table 1: Key requirements for a WRN as defined by Dispena Solutions

	Key requirements	*Example*
1	Reusable for above specified type of use-cases.	The solution must be reusable for the use- cases, which are described in Paragraph 1.2.
2	Save data over the Internet	An individual player (and their trainer) would like to analyze the data over a period of time, and compare these with those of other players. This has as a consequence that the data had to be stored in a database-server.
3	Use one or more sensors, wirelessly, in a sequence.	A football player would like to measure the time he needs for doing a certain exercise, in this case the time of a the slalom is measured, by measuring the motion at the individual football poles, ending with measuring the shooting power. The distance between the sensors, will be 80 meters maximum.
4	Coordinate and synchronize sensors.	Measuring the time during a football slalom has as a consequence that the sensors had to be somehow synchronized. One common clock had to be used in order to measure the time difference between the sensors.
5	Contains a Graphical User Interface	In all cases the application itself should have a graphical interface to visualize the intermediate results.
6	Use Plug and Play, with easy setup.	The required knowledge for setting up the WSN should be minimalized. The system had to work as plug and play/ measure.
7	Easy to maintain.	The maintenance of the WSN should be minimal, No specific experience required.
8	Low cost.	Easy to set-up, easy to maintain and low cost are usually important requirements from marketing and sales perspective.
9	Use existing hardware	The hardware should be everywhere available and standard.

10	Reduce energy consumption	A sensor node consists of a processor, memory, wireless modem and a power supply. It is essential, to have a low energy consumption.

1.4 Research question

The basic research question are:

How to develop a low-cost, reusable, and easy-to-use wireless sensing network ?

What is the best architecture to meet the requirements?

What protocol should be used ?

PROOF OF CONCEPT

The sport example will be implemented and built as a proof of concept as a possible implementation of a WSN network for this use-case.

1.5 Research scope and boundaries

The research scope and boundaries are defined by the research questions as posed in the previous paragraph, and the applications are based on the market opportunities as seen by Dispena Solutions.

1.6 Scientific relevance

This research is innovative and scientifically relevant since it is focused on the current lack of existing low-cost, and reusable Wireless Sensing Network, Chapter 2, Jiang [**1**]).

2 Literature overview

The literature study focused on the research questions: How to develop a low-cost, reusable, and easy-to-use wireless sensing network?; What is the best architecture to meet the requirements?; And what protocol should be used? The literature study is therefore split into:

Paragraph 2.1: Introduction and overview of Wireless Sensor Networks
Paragraph 2.2: The sensor node
Paragraph 2.3: Wireless communication form sensor to base station
Paragraph 2.4: The base station
Paragraph 2.5: Serial Communication Protocol
Paragraph 2.6: Synchronization Protocols

2.1 Introduction and overview of Wireless Sensor Networks

This chapter gives an overview of applications, challenges, problems and solutions for WSNs.

A wireless sensor network consists of sensors to cooperatively monitor physical or environmental conditions. This chapter summarizes two publications i.e. "Survey of applications of wireless Sensor Network using Cloud " by Dash [**1**], and an "Introduction and overview of Wireless Sensor Networks" by Jiang [**3**]. They describe the techniques behind wireless sensor networks in the Cloud.

2.1.1 Applications of Wireless Sensing Networks

Wireless sensor networks are used in different domains throughout the world. Dash [**1**] makes a division into the following application scenarios (see figure 3).

- Transport: The transport monitoring system includes basic management systems like traffic signal control, navigation, automatic number plate recognition, toll collection, etc.
- Military Use: Sensor networks are used in the military for Monitoring friendly forces, equipment and ammunition, Battlefield surveillance, Targeting, biological and chemical attack detection, etc.
- Weather forecasting: Weather forecasting is the application to predict the state of the atmosphere for a future time and a given location. Weather monitoring and forecasting systems include: Data collection, Data assimilation, Numerical weather and forecast prediction.
- Health care: In some modern hospitals sensor networks are constructed to monitor physiological data of patients, to control the drug administration track and to monitor patients reaction.

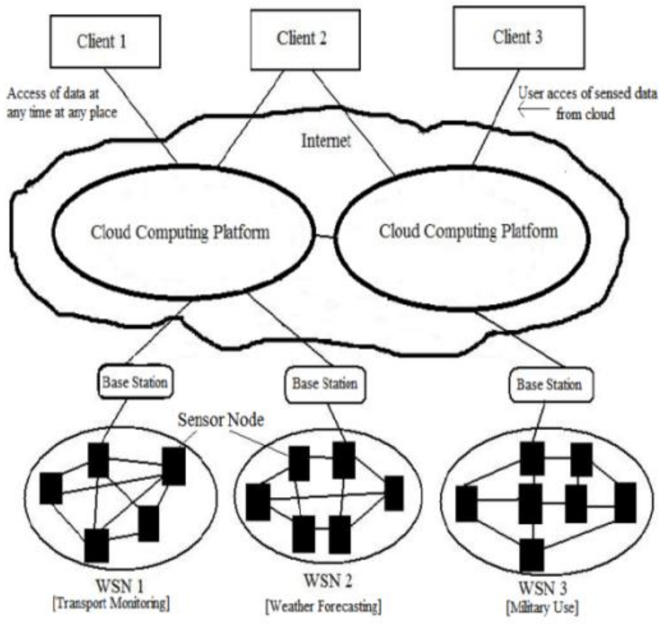


Figure 3: WSN - Cloud Computing Platform

The literature separates wireless sensing networks based on their domains and not on use-case types. The reason for this is the domain dependency of use-cases, as described above.

2.1.2 Elements and topology of wireless sensor networks

Dash [1] describes a wireless sensor network as a system consisting of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure. An overview of important terms described by Dash is:

1. Sensor node: A basic unit in a sensor network, with processor, memory, wireless modem and power supply.
2. Network topology: A connectivity graph with nodes and edges, and links.
3. Routing: the process of determining a network path from resource to destination.
4. Resource: A combination of sensors, communication links, processors, memory, and node energy.
5. Data storage: the storage can be local on the node (where the data is generated), or at a server.

A typical architecture is given in figure 4. In order to overcome the distance between the sink node (or in other literature, often called base node), sensor nodes are often re-sending the information to the next nearest node close to the sink node.

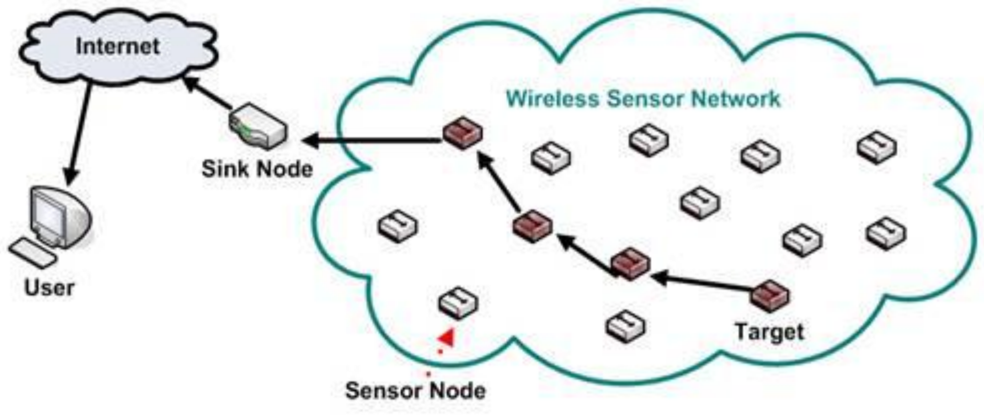


Figure 4: Typical architecture of WSN

Dash [**1**] focuses on the end-user, somewhere in the Cloud, and describes that it makes a lot of sense to integrate sensor networks with the Internet, meaning that data (of a sensor network) should simultaneously be available at any time, at any place. It is pointed out that Cloud computing strategy can help business organizations to conduct their core business activities with less hassle and greater efficiency. Companies can maximize the use of their existing hardware using Cloud computing in combination with wireless sensing.

2.1.3 Problems of wireless sensor networks

Jiang [**3**] provides a list of difficulties in the development of WSNs.

- The limitation of resources of the sensor nodes: Many wireless sensor have limited resources such as a small memory, weak computational capacity, limited energy, and/or a narrow bandwidth.

- The lack of a common architecture: Due to big differences among different applications and domains, many different operating systems, routing protocols and wireless communication approaches have been developed, which is a big obstacle for generality.

- The lack of unified sensors: Existing WSN's use sensor nodes, which are not standardized. For example, there are various type of sensors e.g. for measuring detecting gas, temperature, humidly, or vibration.

Conclusion derived from Dash [1] and Jiang [1]
Jiang [**3**] explains the current limitation of wireless sensor network, i.e. generality and lack of common architecture.

Dash [**1**] shows the upcoming uses of WSN in different domains, this applies the further needs of research on WSNs.

Dash [**1**] uses a ***base station*** (figure 4) as the gateway between the sensor nodes and the internet. However, Dash does not give a definition of the base station. However, it could be derived, from the article that the base station has the role of a collector.

Paragraph 2.2 gives an overview of wireless techniques for connecting the sensor node to a base station. Paragraph 2.3 focuses on the bridge between the base station and the internet.

2.2 Sensor node

A sensor node (figure 5) consists of a basic unit in a sensor network, with processor, memory, wireless modem and power supply (paragraph 2.1, Dash [**18**]).

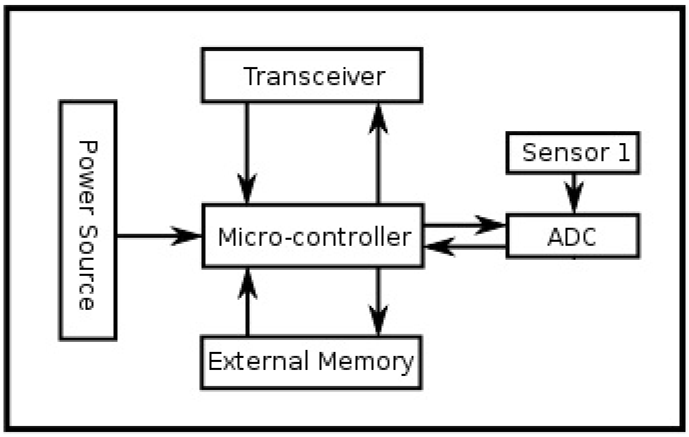


Figure 5: Sensor network

The sensor node is wirelessly connected to the base station.

2.3 Wireless communication from sensor to base station

The type of the wireless modem has consequences for the possible network topology. Figure 6 gives an overview of common network topologies. *However the bus and ring networks are not applicable for WSNs and are only used for wired networks.*

Table 2 gives an overview of common wireless standard networks and the possible network topology for each type of wireless network.

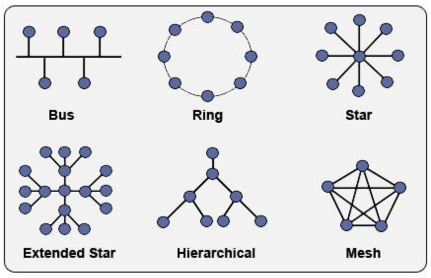
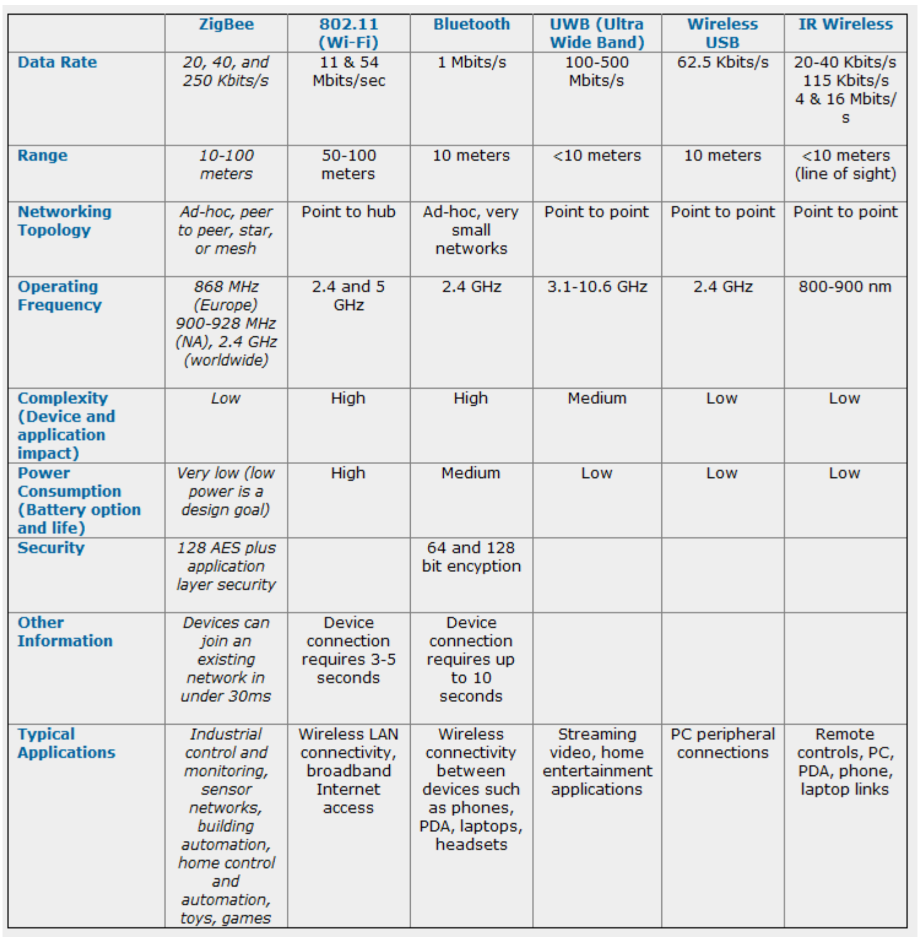


Figure 6: Network protocol

Table 2: Common wireless standards and the consequence.[2]



	ZigBee	802.11 (Wi-Fi)	Bluetooth	UWB (Ultra Wide Band)	Wireless USB	IR Wireless
Data Rate	*20, 40, and 250 Kbits/s*	11 & 54 Mbits/sec	1 Mbits/s	100-500 Mbits/s	62.5 Kbits/s	20-40 Kbits/s 115 Kbits/s 4 & 16 Mbits/s
Range	*10-100 meters*	50-100 meters	10 meters	<10 meters	10 meters	<10 meters (line of sight)
Networking Topology	*Ad-hoc, peer to peer, star, or mesh*	Point to hub	Ad-hoc, very small networks	Point to point	Point to point	Point to point
Operating Frequency	*868 MHz (Europe) 900-928 MHz (NA), 2.4 GHz (worldwide)*	2.4 and 5 GHz	2.4 GHz	3.1-10.6 GHz	2.4 GHz	800-900 nm
Complexity (Device and application impact)	*Low*	High	High	Medium	Low	Low
Power Consumption (Battery option and life)	*Very low (low power is a design goal)*	High	Medium	Low	Low	Low
Security	*128 AES plus application layer security*		64 and 128 bit encyption			
Other Information	*Devices can join an existing network in under 30ms*	Device connection requires 3-5 seconds	Device connection requires up to 10 seconds			
Typical Applications	*Industrial control and monitoring, sensor networks, building automation, home control and automation, toys, games*	Wireless LAN connectivity, broadband Internet access	Wireless connectivity between devices such as phones, PDA, laptops, headsets	Streaming video, home entertainment applications	PC peripheral connections	Remote controls, PC, PDA, phone, laptop links

Let us look at the different options more closely:

[2] *Copied from "Software Technology group:" http://www.stg.com/wireless/ZigBee_comp.html*

- ZigBee: ZigBee is used for high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for personal area networks.
- 802-11 (Wi-Fi): this technology allows an electronic device to exchange data wirelessly (using radio waves) over a computer network
- Bluetooth: Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength radio transmissions) from fixed and mobile devices, creating personal area networks (PANs) with high levels of security
- Ultra-Wide Band: Ultra-wide Band is a radio technology pioneered by Robert A. Scholtz and others, and may be used at a very low energy level for short-range, high-bandwidth communications.
- Wireless USB: **Wireless USB** is a short-range, high-bandwidth wireless radio communication protocol created by the Wireless USB Promoter Group.
- IR Wireless: IR wireless is the use of wireless technology in devices or systems that convey data through infrared (IR) radiation.

Based on the table, possible candidates techniques for connecting the sensor node to the base station are:

- Xbee

  Base Node ←XBee→ Multiple sensor nodes

- Bluetooth

  Base Node ← Bluetooth→ Multiple sensor nodes

- Wi-Fi

  Base Node ← Wi-Fi→ Multiple sensor nodes

2.3.1 XBee

XBee is a brand name of Digi International for a family of compatible radios. It is based on more advanced wireless technologies, and is widely integrated in home automation all over the world. XBee is the radio and ZigBee is the communication protocol for XBee radios.

Compared to Bluetooth, XBee is more efficient in its power management, it has a wider working range, and has a very stable and bug-free protocol, including checksums.

XBee and Arduino

The XBee radio has a range from 40 up to 120 meters (outside). The XBee radio can operate alone, without the use of an Arduino for sensing or forwarding control signals. The XBee has four analog input pins, and eleven digital input/output pins. Since the XBee uses a radio, it needs an internal or external antenna, wire or chip.

The publication “Using Xbee transducers for wireless data collection” by Ayars [**3**], explains the use of Xbee for measuring the physical world. Ayars describes that teachers today have an amazing array of sensors available for use in teaching laboratories; these are often individual sensors that can measure capacitance, temperature, humidity, barometric pressure, GPS co-ordinates, magnet field, orientation in space, and many other physical properties. Ayars describes that the use of Xbee is promising in facilitating measurements for standalone single sensor networks that would not otherwise be possible.

“Xbee Wireless Sensor Networks for Heart Rate Monitoring in Sport Training” from Zulkifli [**4**] gives an example of a sensor (which Dispena Solutions would like to use in their FootballNote application). Heart Rate Monitors are becoming popular, and are widely used in various sports. The principle (figure 7) is simple, a user is wearing the heart rate strap on the chest. A receiver reads the values.

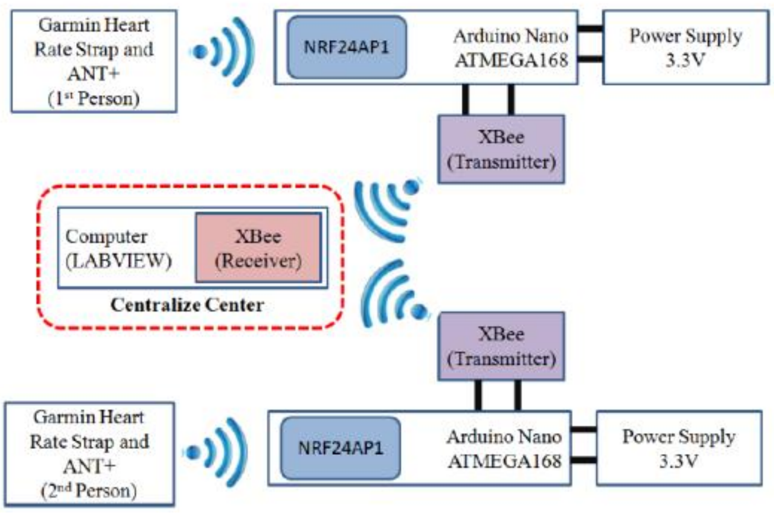


Figure 7: Xbee system for reading Heart Rate

Conclusion derived from Ayars [3] and Zulkifli [4]

Ayars [**3**] describes and tested that the use of Xbee is promising in facilitating measurements that would not otherwise be possible. However, the author writes that there are significant technical details which must be dealt with in order to set up XBee transducers as wireless sensors, since the initialization process of Xbees is not straightforward. However, once the initialization process is done, the Xbees works as expected. The initialization process requires the use of a specific configuration tool[3]. With this tool, the Xbee radios can be updated to the last firmware, and had to be used for configuring the exact address of the radio’s as well setting the (wireless) PAN identification. It is used for different technologies such as: Bluetooth, Wireless USB, XBee, etc.

[3] *In order to update the low-level firmware of Xbee radio’s, the Digi’s configuration tool, X-CTU needs to be used. Note: the program is only available for Microsoft Windows operation systems.*

This should be taken into account during implementation of the Xbee in a project. Ayars indicate that a possible solution could be the use of Xbee radios for wireless sensor nodes.

Zulkifli [**4**] developed a system for measuring multiple heart rates. Looking at the network topology one receiver is used, acting as a base node. The role of this base node, is collecting the data from different sensors and visualize the data.

Figure 8, shows the **Xbee** shield, which allows an Arduino board to communicate wirelessly using Zigbee.

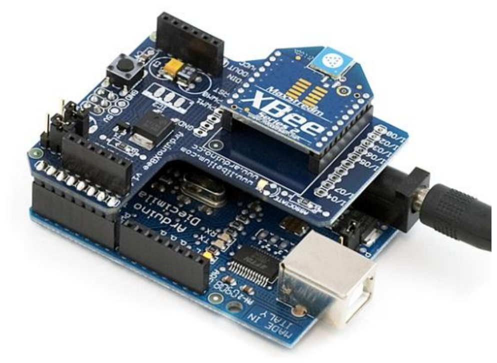


Figure 8: Xbee shield + Arduino

2.3.2 Wi-Fi

W-Fi allows an electronic device to exchange data wirelessly (using radio waves) over a computer network, without needing to plug in an Ethernet cable. Once connected to a wireless router (WLAN), it supports single connection to a router. It could be compared with an Ethernet connection, since it only differs in implementation of the OSI layer one and two. On top of OSI layer three and four, IP, and TCP or UDP can be used for transferring data. An example software implementation of this technique with Arduino uses Client/Server Architecture, where a server has a coordinator role.

Figure 9 shows an example of a Wi-Fi sensor network.

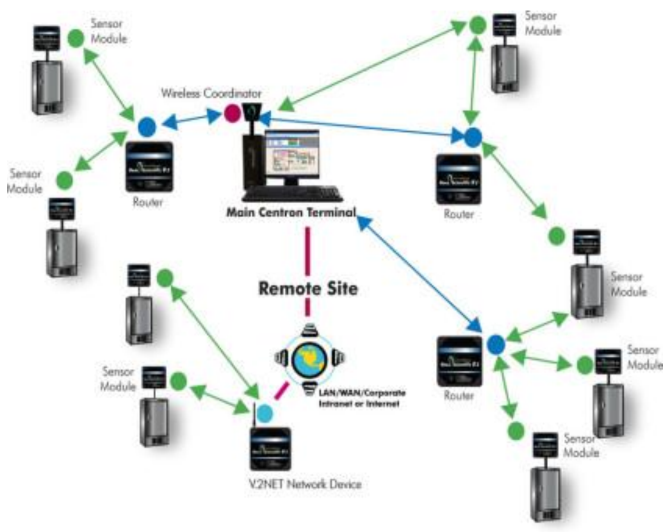


Figure 9: Wireless Sensing network[4]

2.3.3 Bluetooth

Bluetooth is a short-range wireless communications technology used to communicate between devices over a distance of up to about 8 meters. The most common Bluetooth devices are headsets for making calls or listening to music, hands-free kits for cars, and other portable devices, including laptops. Bluetooth is widely used, and embedded in all mobile devices.

Bluetooth networks are like LANs, but are not implemented in the OSI model, because they are only used for point-to-point communication between devices close together. Bluetooth is in fact just a wireless replacement for a serial cable.

For Bluetooth devices to connect to each other so they can work together, they need to be paired. Pairing or bonding means that the two devices are exchanging their passwords or passkeys. Once paired, all of the data that is sent between the two devices is encrypted, meaning that any device that is not paired with the other two is unable to translate the data.

Since the distance between two nodes must be not more than 8 meters, it will not be suitable for the sensor network.

[4] Example of a Wi-Fi sensing network: http://www.reesscientific.com/products-services/wireless-and-wifi-system

2.3.4 *Conclusion*

Based on the literature, Xbee is most suitable for connecting a Sensor Node to the Base station, see table 3. The Xbee technique will be considered for the described problem (see chapter 1).

Table 3: Mobile connectivity with Bluetooth

Technique	***Sensor Node >> Base station***
Xbee	✓ Suitable for a range up to 120 meters.
Bluetooth	Only suitable for small networks, and a distance smaller than 8 meters. A master Bluetooth device can communicate with a maximum of seven devices in an ad-hoc computer network.
Wi-Fi	A Wi-Fi device had to connect to a wireless access point in order to access the Internet. However, an Wireless access point is not available outdoor, therefore a Wi-Fi is not considered as possible solution for these type of use-cases described.

2.4 The base station

There are two types of nodes. One is the normal sensor node deployed to sense the phenomena and the other is a gateway node, this node is to interface the sensor network to the Internet; we call this the base station.

The task of the base station consists for the described problem (chapter 1) of two parts:

- Displaying the data. For this a Graphical User Interface is required, which could be realized using a mobile device.
- A wireless connection to the internet.

As a consequence of this, the research will be focused on wireless networks including a connection with a mobile device.

This chapter describes the research done on connecting a base station to the internet. Depending on the application, the base-station can be connected to the internet using:

- USB streaming

  Mobile ←USB→ ***Base Node*** ←XBee→ Multiple sensor nodes

- Bluetooth

  Mobile ←Bluetooth→ ***Base Node*** ←XBee→ Multiple sensor nodes

2.4.1 *USB streaming*

The connection between a mobile device and a Arduino could be realized over USB. For this Google developed an Android Development Kit (ADK), which makes it possible to connect an Arduino with mobile devices running the Android Operating System. In order to use this functionality an Arduino-ADK board is required. The Arduino-ADK compares with the standard Arduino but it is slightly bigger and has an extra USB port.

The Arduino board acts as an accessory to an Android device. An Android accessory is a physical accessory that can be attached to an Android device. When connecting the Arduino-Adk board to an Android device, the Android device will recognize the Arduino-ADK (as plug-and-play), and through identification the correct Android application will show up.

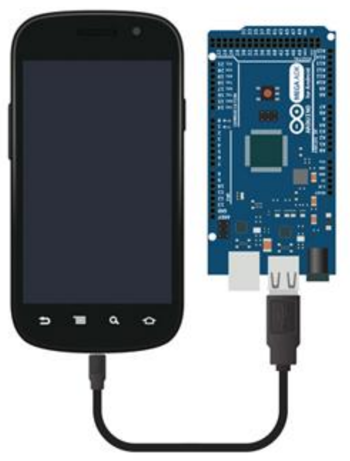


Figure 10: An Arduino-ADK connected to an Android Mobile Phone.

One of the strongest points of this technique, the mobile phone, can be used for running a Graphical User Interface, which is one of the key requirements.

The USB-host capability of the Arduino-ADK makes it possible to communicate (as an accessory) to the mobile device:

The accessory controls the board, detects it and sets up the communication with the Android mobile device, as follows [**5**]:

- Wait for and detect connected devices.
- Determine the devices accessory mode support.
- Attempt to start the device in accessory mode.
- Establish communication with the device.
- Start the required application.

In August 2012, Brandt wrote a thesis about this subject titled: “Efficient data collection using Android ADK in a high velocity, mobile environment” [**6**]. Brandt [**6**] uses the Arduino-ADK connected to a mobile device, see figure 11.

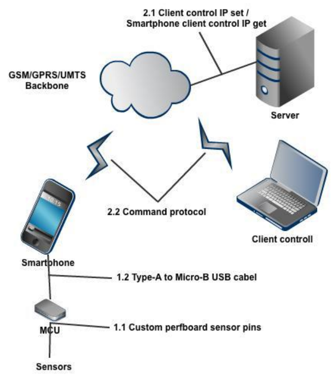

2.1 Client control IP set / Smartphone client control IP get
GSM/GPRS/UMTS Backbone
Server
2.2 Command protocol
Client controll
Smartphone
1.2 Type-A to Micro-B USB cabel
MCU
1.1 Custom perfboard sensor pins
Sensors

Figure 11: Connecting a MCU with a smartphone.

The mobile Android equipment receives the sensor data from the MCU accessory through the USB cable and displays the average of the four sensor on its screen; it either records the data before sending it to the server or directly streams the data to the server.

Conclusion derived from Brandt [6]
Brandt concluded that the Arduino-ADK in combination with a mobile can be used as gateway to the Internet, however needs some more research on the performance. Table 4 shows the compliance requirements of the system, which are moderate.

Table 4: Compliance table of system requirements

Degree	Mobility	Functionality	Performance	Extensibility
Not good				
Moderate	X		X	
Ok		X		
Quite good				X
Excellent				

2.4.2 Bluetooth

Bluetooth is a standard wireless technology for exchanging data over a short distance using radio. It works with mobile devices, creating –as with Xbee- a Personal Area Network (PAN).

As described in paragraph 2.3.3, the phone and a device have to be paired before connecting is possible. The processes of pairing a phone with a device and using it for sensing data is shortly described in this chapter.

The thesis “Design and implementation of a toolkit for the rapid prototyping of mobile ubiquitous computing” [**7**] by Kaufmann focuses on the development of a library for Arduino-Adk (platform), using Bluetooth for connection with a mobile device (figure 12), with the following goals:

- Simple use (beginners).
- Extensible (experts).
- Open (community).

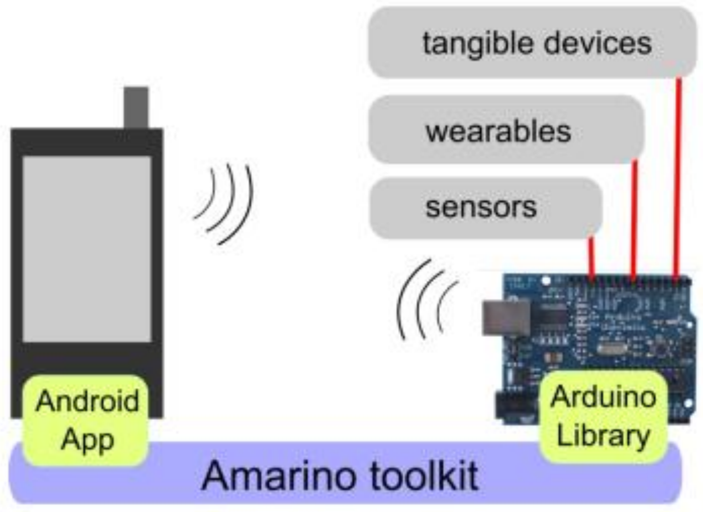


Figure 12: Amarino toolkit

Kaufmann [**7**] describes the process of the development of a library for connecting a Sensor Node (including Bluetooth) with a Mobile Device (figure 12). However, not all mobile operating systems are suitable (Table 5).

Table 5: Mobile connectivity with Bluetooth

Requirements / Platforms	Bluetooth Support	Background Processes	Openness	Accessibility	Low Complexity
Android 2.x	🟩	🟩	🟩	🟩	🟩
Blackberry OS	🟩	🟩	🟩	🟥	🟩
iPhone OS	🟩	🟥	🟥	🟥	🟥
Symbian	🟩	🟩	🟩	🟩	🟥
Windows Mobile	🟩	🟩	🟥	🟩	🟩

Conclusion derived from Kaufmann [7]

Kaufmann concluded that the described WSN is suitable for a connecting a single Bluetooth device to a mobile.

The integration with an iPhone was realized by Alasdair [**8**] in 2012.

2.4.3 Conclusion

Based on the literature on connecting the base station to the Internet, Bluetooth or USB-streaming could be used (table 6). This will be further investigated, when applying the data from the literature to the problem described in chapter 1.

Table 6: Connection of base node to internet

Technique	***Base node >> Internet***
Xbee	
Wifi	
Bluetooth	✓
USB Streaming	✓

2.5 Serial Communication Protocol

Sending information in the network can be realized using a Simple Serial Communication Protocol Based on the Wireless Sensor Network", Liu [**9**], and figure 13.

This protocol can be used for Xbee, Bluetooth and USB Streaming.

In wireless communication systems, the error code rate is high due to external interference. As a consequence, the protocol used for transformation has to be reliable. The core solution creates a frame header including a stream of bytes with a frame end, and an error checksum. The error checksum is used to see, if a package did receive properly. When the checksum is not correct, the message could be resend.

Liu [**9**] describes the following structure for sending a request command (table 7), and receiving data back from a node (table 8).

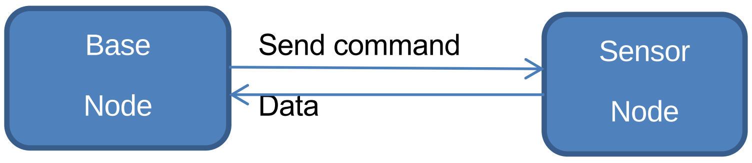
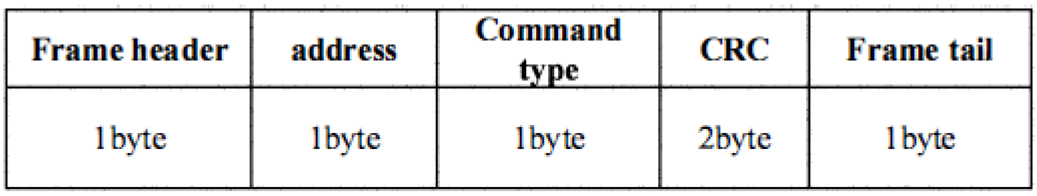


Figure 13: Sending and receiving from base to sensor node

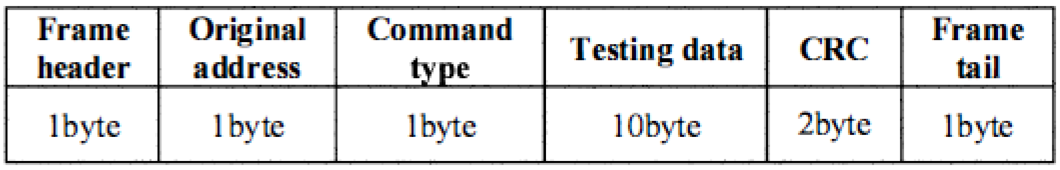
Table 7: Sending command



Frame header	address	Command type	CRC	Frame tail
1byte	1byte	1byte	2byte	1byte

- Frame header: the beginning of the command frame.
- Address: destination address.
- Command type: type of command, for example, reading/writing, etc.
- CRC: Correction checksum.
- Frame Tail: The last byte of the frame.

Table 8: Receiving data



Frame header	Original address	Command type	Testing data	CRC	Frame tail
1byte	1byte	1byte	10byte	2byte	1byte

- Frame header: the beginning of the command frame.
- Original Address: address from the sender.
- Command type: type of data.
- Testing data: the test data.
- CRC: Correction checksum.
- Frame Tail: The last byte of the frame.

CONCLUSION DERIVED FROM LIU [9].

Liu concluded that the Serial Communication Protocol could be suitable for WSNs. However, it had to be adapted for the precise application, since a part is application dependent.

“Beginning Android ADK with Arduino” [5] by Bohmer describes the same protocol for sending data from an Android-ADK to a mobile device.

2.6 Synchronization Protocols

A WSN does not have a common clock. Therefore, a timing protocol is required. This chapter will give an overview of the issues in time, followed by some important synchronization protocols.

Issues on timing

When sending data over the radio from sender to receiver, there will be a delay in receiving.

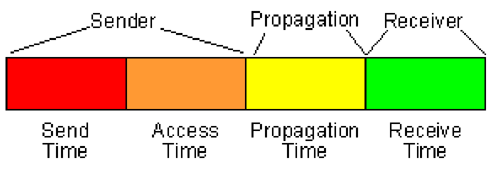


Figure 14: Timing in a WSN

As shown in figure 14, there are four types of delay in a WSN: send time, access time, propagation time, and receive time. The send time is that of the sender needs to construct the message to transmit on the network. The access time is the time, required to access the network. The time for the bits to by physically transmitted on the medium is called the propagation time. Finally, the receive time is the delay in receiving the message. The major problem of time synchronization is not that this packet delay exists, but how to synchronize it.

- To synchronize to global time (figure 15), a node must compute the offset between its local clock and the global time as well as rate at which its clock is drifting slower or faster than global time.

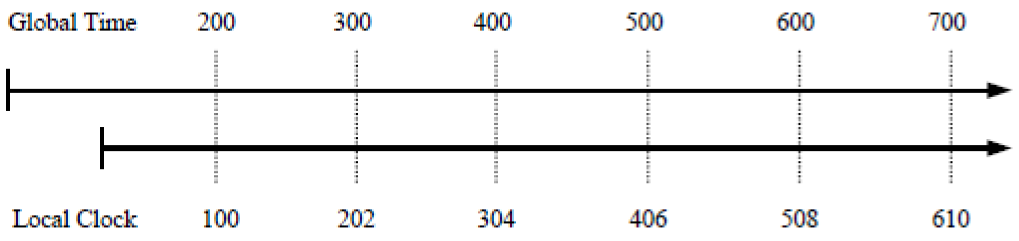


Figure 15: Global Time versus Local Clock

Synchronization protocols

For this the following synchronization protocols were found in the literature:

- Reference Broadcast Synchronization (RBS) is a method in which the receiver uses the physical layer broadcasts for comparing the clocks.

  A third system will broadcast a beacon to all the receivers (sensor nodes). The beacon does not contain time information; it will be only used to compare the relative offsets. The timing is based on when the node receives the reference beacon.

  In a simple form, there are one broadcast beacon and two receivers. The timing packet will be broadcasted to the two receivers. The receivers will record when the packet was received according to their local clocks. Then, the two receivers will exchange their timing information and be able to calculate the offset. This is enough information to retain a local timescale.

- Timing-sync Protocol for Sensor Networks (TPSN)
  This protocol is specific for a multi-hop WSN, where the communication between two end nodes is carried out through a number of intermediate nodes whose function is to resend information from one point to another.

  TPSN is a traditional *sender-receiver* based synchronization network that uses a tree; this tree is used to organize the network topology, where each node is assigned to a level. The level indicates the time in sending and receiving. The construction consist of a root node and all nodes are synchronized to the root in levels. The concept is broken into two phases: the level discovery phase and the synchronization phase. The level discovery phase creates the hierarchical topology of the network wherein each node is assigned a level. Only one node resides on level zero, the root node. In the synchronization phase all *i* level nodes will synchronize with *i-1* level nodes. This will synchronize all nodes with the root node.

- Flooding Time Synchronization Protocol (FTSP)

  FTSP is a variant of the TPSN protocol for a multi-hop WSN, where the latter is bringing down the number of synchronization messages which are sent over the network.

- Variant of Timing-sync Protocol for Sensor Networks (TPSN) for Star-structure networks

  This variant gives some energy reduction for Star-structure networks.
  When the number of synchronization message is reduced, it will give some energy reduction. The described problem (Paragraph 1.2) uses a star-network structure. Therefore, this protocol will be discussed in more detail.

Timing-sync Protocol for Sensor Networks (TPSN) in a Star-structure

The article “Clock Synchronization Method for Star-Structure Wireless Sensor Network” by Li describes a TPSN synchronization protocol adapted for a star-structure. The star-structure is often used in networks, where the distance between the base-node and sensor-node is small.

The standard clock synchronization protocol TPSN requires a high synchronization accuracy, to overcome clock frequency drifting between nodes, a large number of synchronization messages. This means that for a star-structure networks with *n* child sensor nodes a total of 2*n* synchronization packages is required.

Li describes a system for synchronization in star networks where a TPSN broadcasting system is used, to bring down the number of synchronization packages. This protocol will be described in this chapter.

Suppose Node A is sending to Node B at timestamp T1, Node B will receive the signal at timestamp T2. When B is sending a timestamp back at time T3, it will arrive at timestamp T4.

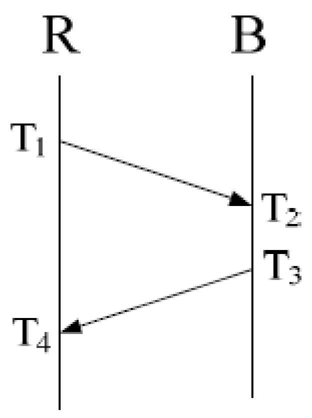


Figure 16: TPSN package exchange between R and B

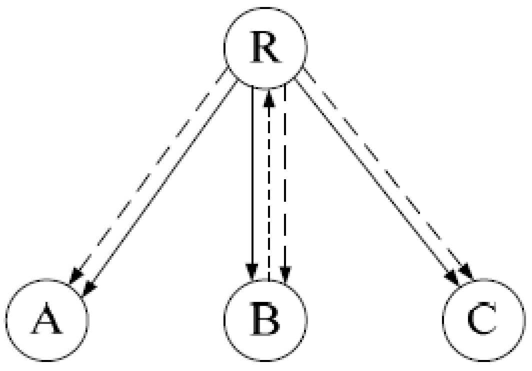


Figure 17: TPSN broadcasting synchronization process

The time can be calculated as follows:

$$T_2 = T_1 + \delta + \varepsilon \tag{1}$$

$$T_4 = T_3 - \delta + \varepsilon \tag{2}$$

The clock deviation between root node R and B can be calculated as follows:

$$\delta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \tag{3}$$

The transmission delay will be:

$$\varepsilon = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \tag{4}$$

The clock deviation (shifting) between the root node is calculated with formula 3, where the transmission delay can be calculated with formula 4.

Using TPSN broadcasting to synchronize the root node, a random node is selected as the respond node. All nodes receive the synchronization signal, but only one node will respond to it, that is in figure 16 node B. Node A and C will record the information and compare it with their own local times. When the root node receives the timing back from note B, it will calculate the transmission and deviation delay. This information is sent to the other nodes, and is used for recalculating the transmission and deviation delay based on the extra offset of B. This way, the number of synchronization packages is reduced to 3.

Using the TPSN broadcasting method, the number of synchronization packages is only 3 packages. If the conventional TPSN is used in star networks, 6 packages are required, and synchronization communication costs will be significantly reduced.

This protocol is tested by Li on star-networks. It can be concluded, that aiming at a star-structure, employing broadcasting synchronization and self-correction of local clock, significantly reduces communication cost on the base of ensuring certain synchronization accuracy.

3 Design

The design of the prototype for measuring the performance of a football player is given in this chapter.

3.1 Use-case

Dispena developed a training program for football (soccer) players, see figure 18. Embedded in this application, Dispena would like to measure the (sport) performance of players, because when the sport performance of individual players is known, an individual training program could be created; the same way a Cooper test is used, which tests the physical fitness of a person, a football player could be tested on his sport performance on the field.

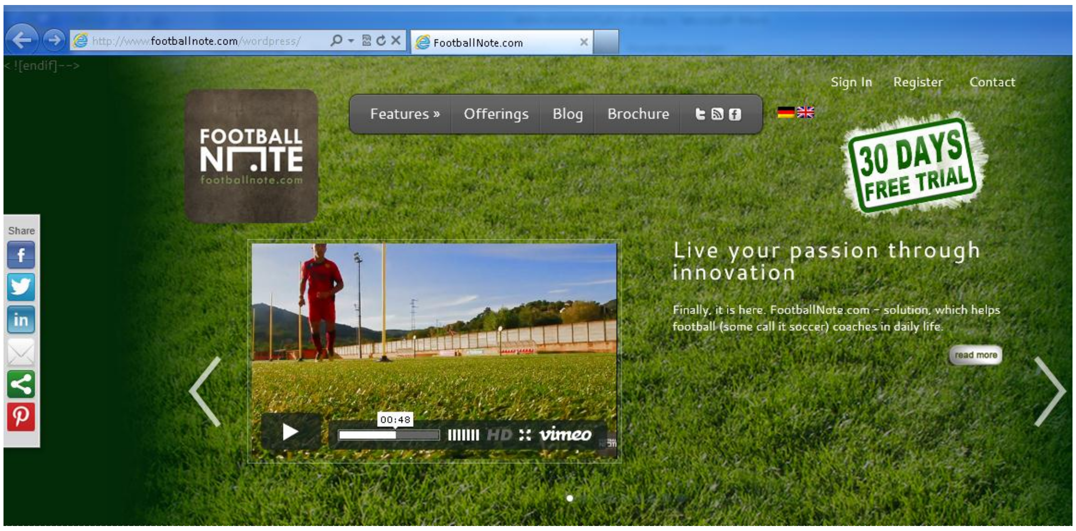

http://www.footballnote.com/wordpress/
FootballNote.com
Sign In
Register
Contact
Features »
Offerings
Blog
Brochure
FOOTBALL NOTE
footballnote.com
30 DAYS FREE TRIAL
Share
Live your passion through innovation
Finally, it is here. FootballNote.com – solution, which helps football (some call it soccer) coaches in daily life.
read more
00:48
HD
vimeo

Figure 18: FootballNote application

The prototype must consist of a system for monitoring a football player during his exercises. During a training, a football player often trains on a slalom continued with a shoot (into the goal).

The prototype should measure the intermediate times during the slalom, followed by measuring the shooting power; see figures 19 and 20. The player should see his results on a screen. All data should be stored in a database in the Cloud, where the FootballNote application can fetch the data.

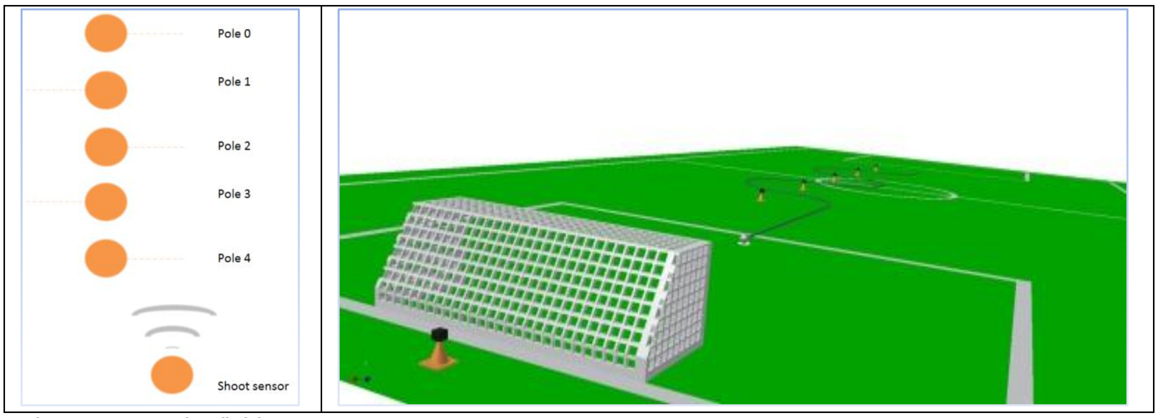


Figure 19: On the field

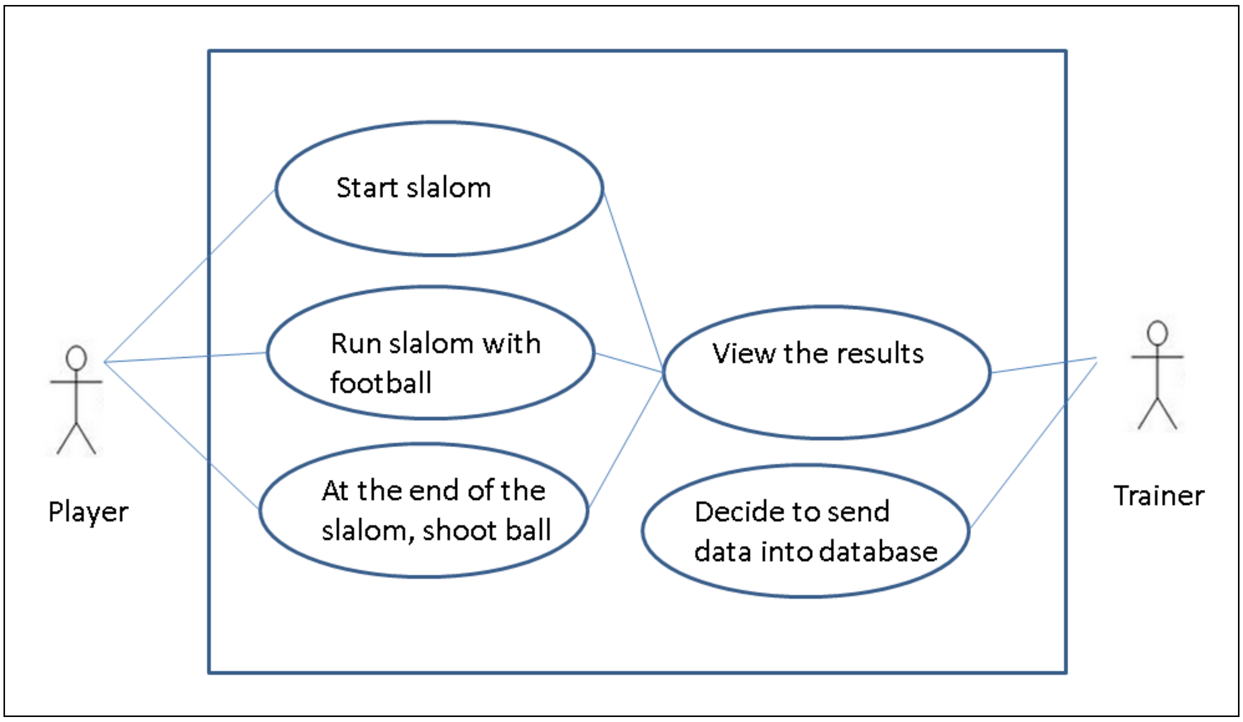

Start slalom
Run slalom with football
At the end of the slalom, shoot ball
View the results
Decide to send data into database
Player
Trainer

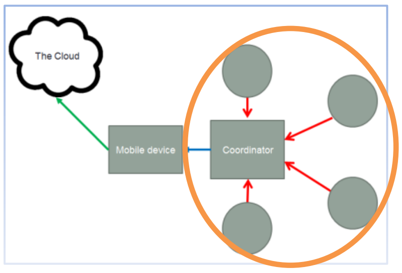
Figure 20: Use-case diagram

3.2 Hardware architecture

Based on the literature described in chapter 2, a decision should be made on the Wireless modems. Since the system should have a Graphical User Interface showing the intermediate results of the football player, a mobile device is required. The decision on which type of mobile device and which OS will be used has a direct influence on the software development. The pro and cons of different solutions will be given in this chapter.

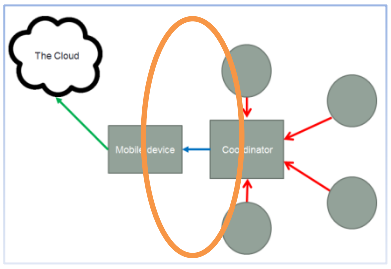
The network can be split into two parts, namely the part connecting the multiple sensors to the base node, and the connection between the base node and mobile device.

- ***Base Node*** ←XBee→ Multiple sensor nodes



Concerning the WNS network for connecting multiple sensor to the base node, the best suitable solution is using Xbee. The two most common RF radios that are available from Digi are the Series 1 and Series 2 XBee. The Series 1 and Series 2 modules are quite similar, but selection of a module should be based upon application specific needs. In this case, where a star network is required, the Series Xbee radio is needed.

- ***Mobile*** ←USB or Bluetooth→ ***Base Node***



For the connection between the base station and the mobile, there are two possible options, namely Bluetooth and USB. The (extra) costs of a Bluetooth module are approximately € 50. Using Bluetooth, there is no limitation in the type of Operating System on the mobile device, since Bluetooth is widely supported. However, this means extra costs (which is in contradiction with the key-requirement “low-cost”), and requires a specific setup on the mobile device, which is in contradiction with the key requirement “plug-and-play”. When USB would be used, this has as consequence that the mobile device should run the Android OS. When connecting the base station (an Arduino-ADK board) to an Android device, the Android device will recognize the board (as plug-and-play), and through identification the correct Android application will show up. This makes the solution “plug-and-play”. However, currently not all mobile devices running the Android OS support the USB mode. Secondly, an Android OS version 2.3 or above is required. Currently, most devices are supplied with version 4.0, so the latter should not be an issue in the future.

Based on these pros and cons the system will consist of:

- Arduino + Xbee series 2 radios for the sensor nodes.
- Arduino ADK and USB interface for the base node.
- A mobile device running the OS Android. For the development, I have chosen the HTC desire S running Android OS version 2.4.

Design costs

In table 9, the costs of each individual product is given. The prices are based on an average of different suppliers, and are a mend as indication.

Table 9: Prices of Arduino parts

Product	Costs
Arduino	$30
Arduino-ADK	$70
Arduino-Shield (optional)	$15
Xbee series 2	$25
Xbee series 2 – Pro	$38
Bluetooth	$60
Wi-Fi (called: WiFly)	$35
USB	$3

This are the costs for prototyping and or for individual projects. The Arduino – Uno costs around $30, however, the chips itself costs around $3 and easily available at distributors. The difference in cost of the chip in compares to the Arduino – Uno prototype board, is the programming facility (for an Arduino the programming is done through the UART requiring an RS232 to USB converter).

The connection with the base station could be realized using Xbee series 2. This is the cheapest solutions to realize a WSN network. The connection with the mobile device (which runs the Graphical User Interface) can be realized using USB or Bluetooth. Because of the low price is this the best solution, however not all mobile devices support a USB connection.

The minimum price of a mobile device running the Operating System Android and supporting USB is around $150.

As an alternative to the Arduino another electronic board could be chosen. A comparison is made in table 10.

Table 10: Prices of alternative boards[5]

Product	**Price**	**Pros and cons**
Arduino-Uno	$30	Arduino is an open-source electronics proto-typing platform based on flexible, easy-to-use hardware and software.
BeagleBone	$90	The BeagleBone from Texas Instruments is the lowest-cost model in the company's BeagleBoard line of prototype boards. It has a powerful processor, an Ethernet port built into

[5] *From: http://www.techhive.com/article/257343/noteworthy_alternatives_to_arduino.html*

		the board, and a MicroSD card slot.
Teensy/Teensy++	\$20-\$30	The Teensy and the Teensy++ are ultrasmall boards. The Teensy comes with 25 I/O pins and the Teensy++ has an impressive 46 I/O pins, enabling either one to support a wide range of peripherals. each also has an on-board USB port, to simplify the task of uploading programs.
Pinguino	\$35	Since the Pinguino is the same shape and size as the Arduino Uno, it can use the shields available for Arduino boards. Unfortunately, this does not guarantee that they will work--and the Arduino development team do not support the Pinguino. A positive point is that it includes an onboard lithium-ion battery charger.
MSP430 Launchpad	\$5	The MSP430 Launchpad, an inexpensive microcontroller from Texas Instruments, emphasizes expandability and ease of use. This board does currently not support Xbee.

For a WSN for a single use-case (no series production), the Arduino-Uno, or as an alternative the Pinguino, could be used. Since, it is not sure of the Pinguino supports the Xbees, the Arduino has been chosen for prototyping.

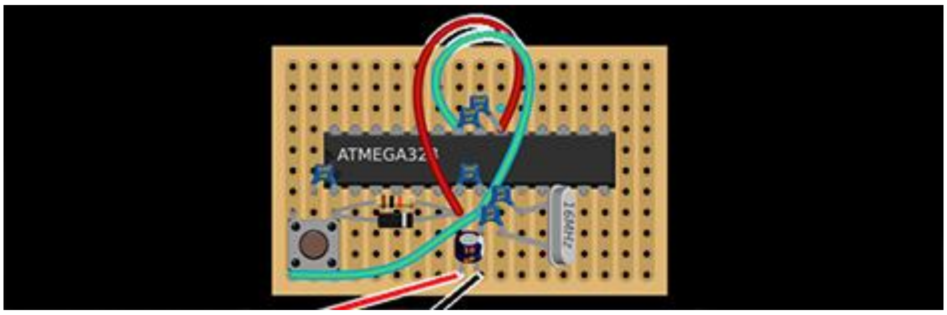


Figure 21: MCU

Once the prototyping is realized, the single microcontroller chip can be used standalone. Figure 21 shows the basic circuit needed, which consists of some resistors, a capacitor and a 16 MHz crystal. It is a bare-bones system, but once built and programmed, it provides all the functionality of an Arduino.

3.3 Measurement

With this test system for FootballNote a player's exercises can be monitored. The measurement consists of the intermediate times realized at points in a slalom followed by measuring the shooting power. The player can see his results on a screen. All data are stored in a database in the Cloud, where the FootballNote application can fetch the data. The process is described table 11. The BPM (Business Process Model) is given in figure 22, and gives an overview of the tasks for each component, split into the task of the player, the sensor nodes, and the FootballNote Graphical User Interface.

Table 11: Main actions of the measurement

Step	*Action*
1	Select a player.
3	Start running (Player).
4	Measure time at slalom pole 0.
5	Measure time at slalom pole 1.
6	Measure time at slalom pole 2.
7	Measure time at slalom pole 3.
8	Measure time at slalom pole 4.
9	Measure speed of ball.
10	Display result of single player.
11	Send values into the database.

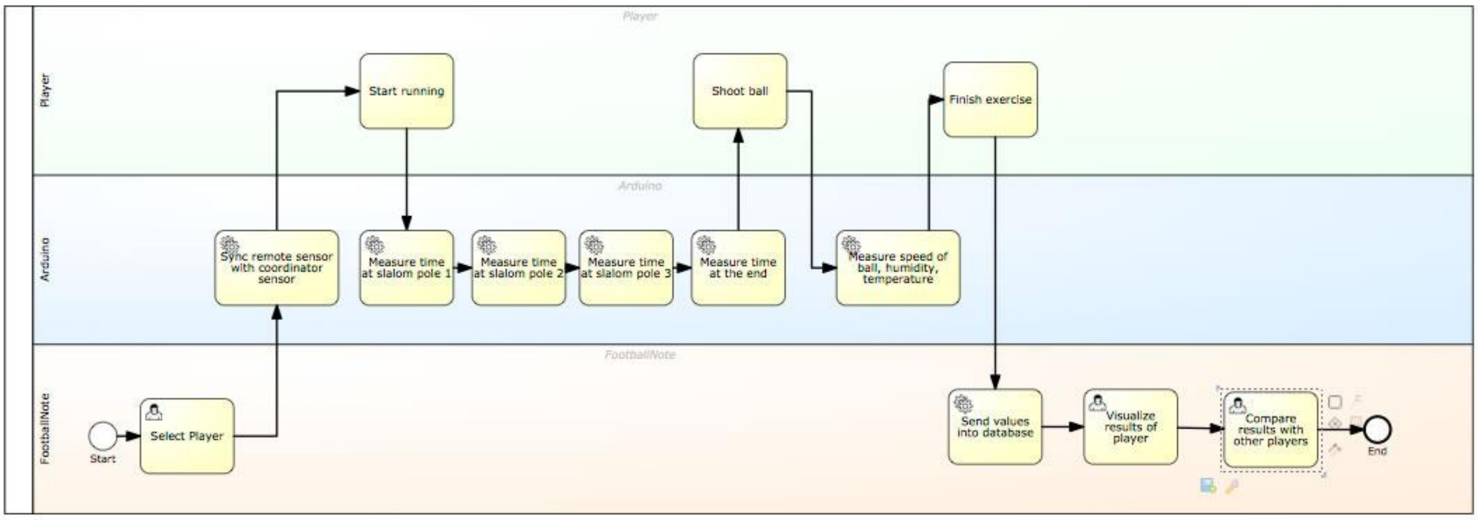

Player
Arduino
FootballNote
Start running
Shoot ball
Finish exercise
Sync remote sensor with coordinator sensor
Measure time at slalom pole 1
Measure time at slalom pole 2
Measure time at slalom pole 3
Measure time at the end
Measure speed of ball, humidity, temperature
Start
Select Player
Send values into database
Visualize results of player
Compare results with other players
End

Figure 22: Business Process Model

3.4 Hardware design

Let us first consider the hardware design and configuration.

As shown in figure 22, movements must be sensed at five places. After the slalom, the shooting power is recorded. To measure the exact timing between the sensoring moments (one to five) a motion sensor is used. At the end of the slalom the shooting power is measured. This can be realized using a radar sensor. This measurement system needs a lot of synchronization, with a common clock. The clock in the coordinator (Arduino-ADK) will be used as synchronizer, and will handle the timing. The Coordinator is connected to a mobile device. As the application is being used in an open field, an Android mobile is used for connection to the internet through 3G. A Graphical User Interface –showing the results- is provided on a mobile device. The data are saved in the Cloud (and in the future embedded within FootballNote).

3.4.1 *Motion sensor*

In order to sense the motion a Passive Infrared sensor can be used.

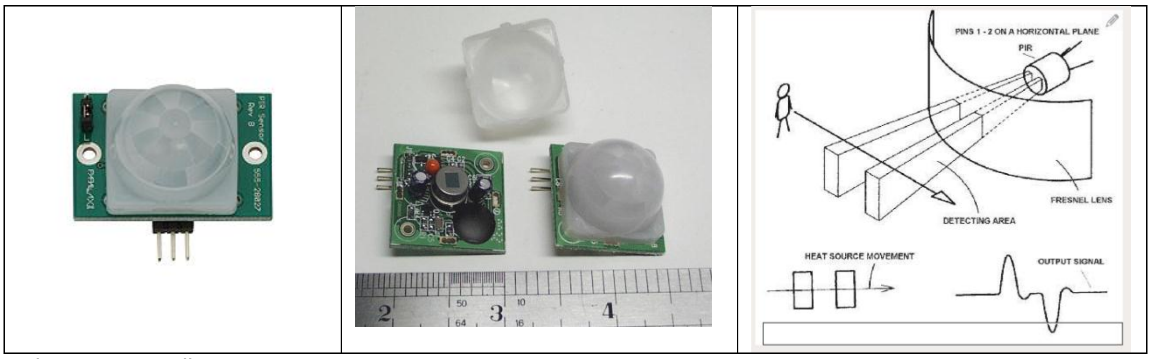


Figure 23: Parallax PIR sensor

A PIR (Passive Infra-Red) Sensor (figure 23) is a pyro-electric device that detects motion by measuring changes in the infrared (heat) levels emitted by surrounding objects. When motion is detected the PIR Sensor outputs a high signal on its output pin (figure 23). The PIR sensor can detect a person up to ~10 meters away, in the reduced sensitivity mode. The principle of motion sensing is shown in figure 23. When a warm body (like that of a human or an animal) passes by, the output signal will change. The Arduino detects the change in pulse height.

3.4.2 *Speed sensor*

An X-Band Motion Detector (see figure 24) can measure the speed of motion.

Figure 24: Parallax X-Band motion detector

An X-Band Motion Detector operates in the X-band frequency, at 10,525 GHz and indicates detected movements with oscillations in its high/low output.

The X-Band Motion Detector's sensor is a common ingredient in security systems and automatic door openers, and can detect movements in a room, yard, or even on the other side of a wall. Sensitivity is manually adjustable with a potentiometer, offering direct line of sight detection from roughly nine meters. The X-Band motion sensor also detects the speed of the motion.

3.5 Software protocols

In order to synchronize the clock between the sensor node and the base node a synchronization protocol is required (Paragraph 3.5.1).

Sending information from the Sensor Node and the Base node can be realized using a Serial Communication Protocol; however the protocol should be adapted for the described problem (Paragraph 3.5.2).

3.5.1 Clock synchronization.

The system does not have a common clock. Therefore, a timing protocol is required. Based on the literature, a solution can be found in implementing the TPSN (paragraph 2.5) for star networks.

3.5.2 Protocol

Sending information in the network can be realized using a Simple Serial Communication Protocol Based on the Wireless Sensor Network", Liu [**9**]. This protocol needs to be adapted for the given situation. The sensor node needs to send the timing information to the Android device. For this, a special message code must be created, like:

1-byte Header H	2-byte Type Low byte	2-byte Time Low byte	3-byte Time High byte	Checksum

The type field is given the type of data.

- A to E : Means motion detected at sensor (pole) 0 to 5.
- S: Speed of motion
- Time speed (byte 2 and 3)
  The time will be send in milliseconds and is defined as integer. Since an integer is bigger than one byte, it is split up into 2 bytes.
- Checksum

The Arduino is using the Arduino IDE, which makes use of the data type "signed byte" for streaming. The data type available in the programming language Processing (for the GUI) only supports the data type "unsigned byte". Therefore, a conversion is required.

3.6 Summary

For the proof of concept a test system for FootballNote is designed. Each individual sensor sends a signal to the coordinator when motion occurs. The coordinator handles the timing, and saves the time of arrival of a signal. The coordinator sends the times to an attached Android mobile device. An application runs on the mobile device, displaying the results, and posting them to the Cloud using PHP.

The complete test system for the FootballNote application consist of:

End devices
Motion sensors and Speed sensor networks act as end-devices, they are standalone sensing systems, which send their outcome through an Xbee radio to the coordinator. Xbee radios are attached to an Arduino. A motion sensor is connected to the analogue input of the Arduino.

Coordinator Unit
An Arduino-ADK needs an Xbee radio which is configured as coordinator for synchronizing and timing. The Arduino-ADK is physically connected (using a USB cable) to an Android device.

User interface
An Android tablet or smart phone is used showing the collected data on the movement and speed of players, and sending the values to the Cloud.

Protocol for communication
The communication protocol between the end devices and the Arduino-ADK have been defined as well the protocol between the Arduino-ADK and the Android device (graphical user interface).

4 Implementation

In this chapter, the implementation of the prototype for measuring the performance of a football player is given.

Paragraph 4.1: Motion sensors.
Paragraph 4.2: Speed sensor.
Paragraph 4.3: Base station.
Paragraph 4.4: Tests.

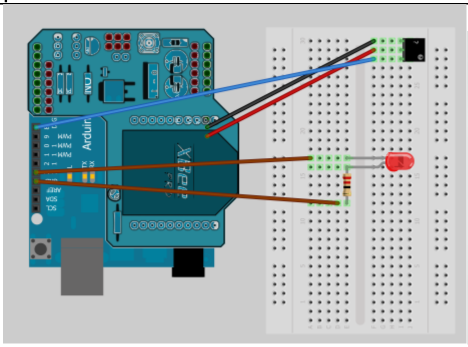
4.1 Motion sensor

The design was simplified, because of the long time required to fully implemented the TPSN synchronization protocol. The task sends a signal to the Arduino-ADK (the coordinator) when motion is detected, the implementation is explained in table 12.

Table 12: Motion sensor - implementation

Task	Send a signal to the Arduino-ADK (the coordinator) when a motion is detected.
Hardware configuration	Arduino Uno; Xbee shield; Xbee; 1 LEDs; 1 transistors (22Ohm); PIR sensor from Parallax[6]
Code	The end devices detect motion. The sensor is connected to an input pin on the Arduino. The function: digitalRead, reads the value from a specified digital pin, either <u>HIGH</u> or <u>LOW</u>. When a motion is detected a signal will be sent to the coordinator like this:

	`…`
	`  val = digitalRead(inputPin);  // read input value`
	`  if (val == HIGH) {            // check if the input is HIGH`
	`      // Motion detected!`
	`       Serial.print('A'); // send a header character`
	`…`

	When the coordinator receives an ‘A’, the coordinator knows that motion happened at place 1. When the coordinator receives a ‘B’ comes from place 2, etc.
Electronic circuit	

6 *The sensor can be ordered at:* *http://www.parallax.com/tabid/768/ProductID/83/Default.aspx*

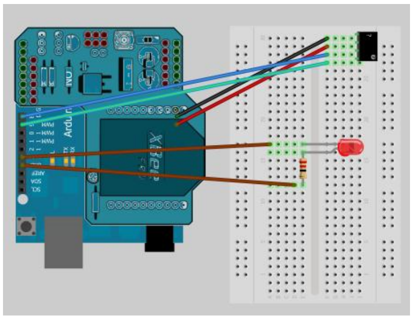
4.2 Speed sensor

The speed sensor will sense the speed of the ball and send the time to the coordinator; the implementation is explained in table 13.

Table 13: Speed sensor - implementation

Task	Sensing the speed of motion; Sending data (motion speed) to the Arduino-ADK (the coordinator).
Hardware configuration	Arduino Uno; Xbee shield; Xbee;1 LEDs; 1 transistors (22Ohm) XBand motion sensor from Parallax[7]

Code	This end device measures the acceleration of an object, in this particular case: shooting power. The sensor is described in paragraph 1.8.2. In this case, the data on the measured speed must be sent to the coordinator as well.
	`Serial.print('S');                    // Send message`
	`Serial.write(lowByte(endcount));      //speed lowpart`
	`Serial.write(highByte(endcount));     //speed highpart`

Electronic circuit	

[7] *The sensor can be ordered at: http://www.parallax.com/StoreSearchResults/tabid/768/txtSearch/X-band/List/0/SortField/4/ProductID/606/Default.aspx*

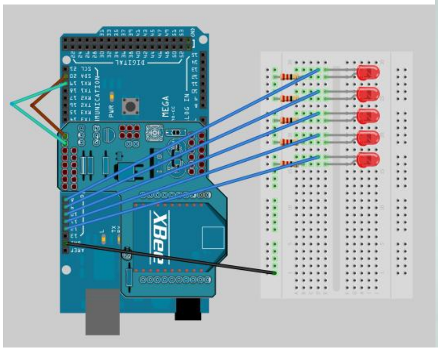
4.3 Base node

The base node has the task of synchronization of the sensor data and sending it to the mobile device. The implementation is explained in table 14.

Table 14: Base node - implementation

Task	-Receive signals when a motion occurs in Arduino 1 to 4. -Receive data on the speed of motion from Arduino 5. -Timing -Send the values to the mobile device.
Hardware configuration	Arduino-ADK; Xbee shield;Xbee;5 LEDs;5 transistors (22 Ohm)
Code	The coordinator must keep track of the time when a signal arrives. The time library creates the possibility to keep track of time. The Arduino-ADK uses a quartz crystal for timing, but does not have a battery to remember the time when power is switched off. The function which must be used is, called: millis(). The function returns the number of milliseconds from the moment the program has started. Code 18 shows a part of the code concerning the timing.

	int startTime =0;
	Int t0=-1; //starting timer
	Int t1=-1; //time place 1
	Int t2=-1; //time place 2
	Int t2=-1; //time place 3
	byte incomingByte = 0;
	
	void loop() {
	  // make sure everything we need is in the buffer
	  if (Serial.available() >= 2) {
	    incomingByte = Serial1.read();
	    // look for the start byte
	…
	    if (incomingByte == ‘B’) {
	          startTime = millis();
	           t1=millis() - startTime;
	
	          // Sending the time that motion happened in place B (t1)
	          acc.print('H'); // send a header character
	         acc.print('B'); // send a header character
	         acc.write(lowByte(t1)); // send the low byte
	         acc.write(highByte(t1)); // send the high byte
	
	    }

Electronic circuit	

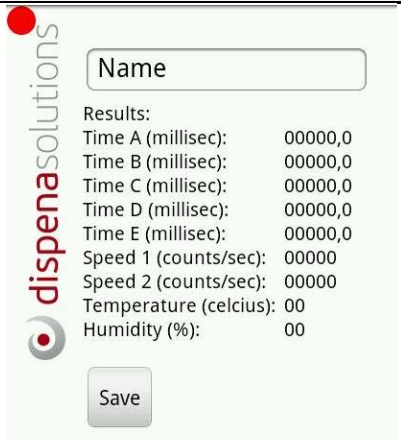
4.4 GUI

The code for the Graphical User Interface is written in Processing. The reason for choosing the programming language Processing is that it is fully integrated within Arduino. The implementation is explained in table 15. The programming language Processing is explained in Appendix C.

Table 15: GUI - implementation

Task	Connect the Arduino-ADK with the mobile device. Run a Graphical User Interface with the player's results.
Code	A mobile Android device is attached to the Arduino-ADK board. It will show the results of the players, and send the values to the Cloud. Part of the code [5] is outlined.

```
void receiving()
{
  // read the header and two binary *(16 bit) integers:
  if ( myPort.available() >= 4)  // If at least 5 bytes are available,
  {
    if( myPort.read() == “H”) // is this the header
      if( myPort.read() == “A”) // is this the header
     {
        value1 = myPort.read();                          // read the least significant byte
        value1 =  myPort.read() * 256 + value1;   // add the most significant byte
        t0 = value1;
     }
      if( myPort.read() == “A”) // is this the header
     {
        value1 = myPort.read();                          // read the least significant byte
        value1 =  myPort.read() * 256 + value1;   // add the most significant byte
        t1 = value1;
     }
      …
  }
}
```

Interface	

Name

Results:

```
Time A (millisec):        00000,0
Time B (millisec):        00000,0
Time C (millisec):        00000,0
Time D (millisec):        00000,0
Time E (millisec):        00000,0
Speed 1 (counts/sec):     00000
Speed 2 (counts/sec):     00000
Temperature (celcius):    00
Humidity (%):             00
```

Save

4.5 Tests

For the proof of concept a test system for FootballNote is implemented; the results can be found in the next paragraphs.

4.5.1 Testing the Xbee network

The Xbee is going into sleep mode during the measurements. The documentation on the Xbee describes the possibility to use the Xbee in a non-sleeping mode. During measurement, I conclude that this mode does not work as it should. To solve this problem, I have found, the 'Hibernate' mode can be used, and by connecting the input pin (for hibernate) with the ground using a wire.

There are different types of Xbee radios Series 2, the standard version and a pro version. The tested FootballNote system uses the Xbee Series 2 Pro. The power consumption was significant higher than that of the standard version. When testing with the standard Xbee Series 2 the power consumption is still too high. A solution can be found in:

- Sensors are to be used in cycles of sleeping and waking.
- Coordinator sends a broadcast to the sensors to wake them up.
- During the whole measurement period the sensors are kept awake.
- After the measurement, the coordinator sends a broadcast to the sensors. Sensors are going back into cycles of sleeping and waking.

4.5.2 Testing the Motion sensor

The motion sensors (PIR sensors) work fine and accurately enough for this type of use-case. However, starting up the sensor takes approximately 5-10 seconds before the sensor is ready. This is due to the setup function of the sensor itself. In order to use the motion sensor, the magnifying glass, is partly taken away, to make it sense in a line (and not in an angle).

4.5.3 Testing the speed sensor

The X-Band Motion Detector is used for speed sensing, however it only works up to 10 kilometers per hour. The shooting power of a ball is at least 100 kilometers per hour. Therefore, more dedicated hardware must be developed. Details concerning the development of a better sensor can be found in appendix D.

4.5.4 Testing the synchronization of the exercise

The system is tested on the football field with professional football players. The system performs as expected, except for the speed sensor. The measurement time of a player doing a slalom was maximally 4 seconds. The coordinator is able to measure the time between the motions, however not accurately enough, because of the time delay in the network. In order to increase the accurately, the TPSN protocol had to be implemented.

4.5.5 Testing the battery consumption

The power consumption is too high, the reason for this is the ineffective way the Xbees are programmed. In the final product a rechargeable battery will be applied.

4.6 Conclusion

The results shows that the prototype works only partly for the football. The speed sensor need to be redesigned. The Xbees had to be programmed in such a way, that they are only in communication mode during a measurement, this will bring down the battery consumption. In order to increase the accurately of the timing, the TPSN protocol had to be implemented.

5 Conclusions

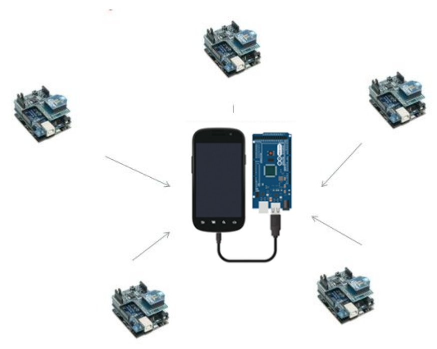
How to develop a low-cost, reusable, and easy-to-use wireless sensing network ?

The proof of concepts shows a system for standardizing WSN, demonstrating the possibility to build a case specific, low-cost and reusable wireless sensing network for the Sport.

In principle the prototype, the development of a system for monitoring an individual football player, could be used for other sport-activities, when using the same concept of measurement.

What is the best architecture to meet the requirements?

The following network is used successfully to realize an application independent wireless sensing network, which is a combination of Xbee radios, USB, and 3G/Wi-Fi.



The system uses a mobile device running the Android Operating System. As an alternative, Bluetooth could be considered, which makes the system compatible with other operating systems, since most of the systems have Bluetooth embedded.

However, this has a consequence for the key requirement “plug-and-play”-mechanism, since this requires some initialization on the mobile device.

Wireless USB could in future be considered as a possible standard. Wireless USB performance is 480Mbps at 3 meters and up to 110Mbps at 10 meters.

How to design a common protocol for sending sensor data?

The standardization is realized using a protocol for sending and receiving data. The coordinator uses a list of sensors, and can easily be adapted with more and different types of sensors; this makes the system extendable. The system is created for football, but should be easily adaptable for other sports. This means that the developed system is applicable for application independent measurements. Furthermore, the same system can be adapted for industry, since it is not unique for sports.

Does the prototype meets the key-requirements?

In table 16 the key-requirements are discussed.

Table 16: Key requirements in the proof of concept

	Key requirements	Proof of concept
1	Reusable for above specified type of use-cases.	The proof of concept works partly for the described problem. However, in order to make it applicable for industry further research needs to be done. As a part of the research for Dispena, the system is extended with some other basic sensors. This forms the start of a prototype for a shoe manufacturer[8].
2	Save data over the Internet	The data could be saved on the internet. However, no further research is done on the method of posting the data into a database.
3	Use one or more sensors, wirelessly, in a sequence.	During measurements, it should be noted, that the measurement could fail, when one sensor is not reacting in the sequence. In order to avoid this, the system should check if all sensors are active before the measurements start.
4	Coordinate and synchronize sensors.	The time delay of the system is not taken into account. In order to use this system the clock synchronization method for star-structure wireless sensor network could be used for this. This system is described in paragraph 2.6.
5	Contains a Graphical User Interface	The concept makes use of a Graphical User Interface running on an Android mobile device. The system will only work with an Android device; in order to make it available for other mobile devices, Bluetooth can be considered.
6	Use Plug and Play, with easy setup.	The prototype works as expected, as plug and play. Since, once connecting the mobile to the base node, the application is automatically launched on the mobile. For this no further software is needed.

[8] An explanation of this shoe factory use case can be found here: http://youtu.be/2JDM3w0sHIA

		When the Arduino-Adk is connected to the mobile device, it cannot be recharged, since it makes use of the same USB port.
7	Easy to maintain.	When the prototype comes into production, the battery place should be reconsidered. In the prototype the battery is inside the box, which is not handy when the battery should be replaced.
8	Low cost.	The system is low-cost. Price reduction can be realized by using the single chip, instead of the Arduino-Uno board. See paragraph 3.2.
9	Use existing hardware	The prototype makes use of existing hardware. However, for specific measurements (like measuring the speed of a football) the existing hardware fails. For this dedicated hardware needs to be developed (see Appendix D).
	Energy consumption	The prototype is ineffective as far as energy consumption is concerned. Since the Xbee radios are continuously in the active mode, even when there is no active measurement. A solution can be found in setting the radio's only in active mode, during a measurement. Because of the high energy consumption, the Pinguino board could be considered as an alternative of using the Arduino-Uno board, since it has a lithium-ion battery charger onboard. See paragraph 3.2

Further developments

This research has led to a collaboration with Dispena for further developed, and to realize application independent solutions for WSN.

After the research, Dispena was able:

- to estimate the needed investment to design and build own sensors.
- to offer an affordable proof of concept to potential customers to implement a measuring process.
- to build a new software to automatically configure the sensors for any kind of measuring process, especially in the manufacturing industry.

The lessons learned from the first prototype resulted in a use case independent design for Dispena, and have led to further development, which will be described in chapter 6.

6 Recommendations

Based on this research, Dispena would like to realize this system as a commercial product. Based on this research, some next step and further suggestions are described hereafter. The further development is not part of this research, and are therefore only recommendation for further analysis and research are given.

Network

The system uses a Graphical User Interface running on a mobile device running Android. More openness and standardization could be realized if the Graphical User Interface would be platform independent. Possible platform independency is described in table 17.

Table 17: Platform independency

Platform	*Possible solution*
Mobile device running Android OS	USB Bluetooth
Mobile device running iOS	Bluetooth
Mobile device running Windows 8	USB, Bluetooth not supported yet
Standalone PC running Windows 7	USB, Bluetooth
Standalone computer running macOS	USB, Bluetooth
Standalone computer running Linux	USB, Bluetooth

The connection between Arduino-ADK and the mobile device is currently realized using the Arduino-Adk board as an accessory. However, not all Android devices support this mode. It is usually not clearly described whether a mobile device supports this connection. Alternatively, it is be possible to directly implement it as a micro-bridge[9].

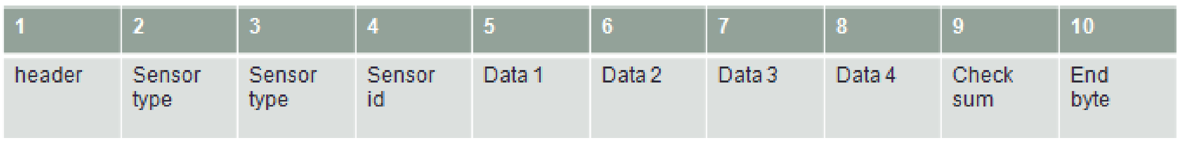
The football application uses a very small protocol for sending a stream of data. Based on “A Design of Simple Serial Communication Protocol Based on the Wireless Sensor Network” [17], the protocol had to be extended.

[9] See: http://code.google.com/p/microbridge/

New Protocol

In order to make it extendable, the sensors should be selectable. Currently the prototype works with one type of exercise. The next step is, to choose an exercise and sensors, a possible new protocol is described in table 18.

Table 18: New protocol



1	2	3	4	5	6	7	8	9	10
header	Sensor type	Sensor type	Sensor id	Data 1	Data 2	Data 3	Data 4	Check sum	End byte

- Header: start of the data stream
- Sensor type (byte 2 and 3): a unique sensor type name
- Sensor id: a unique number
- Data bytes 1 to 4: the sensor data
- End byte: giving the end of the data stream;

XML

In order to store the data in an efficient way, further research could be done on streaming the data in XML format. An example of this is given in table 19, where the temperature is written in an XML file.

Table 19: XML output example

```
<data-sensor>
 <type name="TE" idd="1"/>                                   //name of the sensor + idd  (combination is unique)
 <sequence number="1">                                         //sequence measurement number
 <data value="25.0"/>
 <time hour="16" minutes="30" seconds="38"  milliseconds=""/>     //or using a timestamp
</data-sensor>
```

Data-Synchronization

A protocol for Clock-Synchronization for Star-Structure Wireless Sensor Network should be implemented for timing; therefore, the protocol described in Chapter 1 could be used (Timing-sync Protocol for Sensor Networks (TPSN)).

7 Bibliografie

[1] S. K. Dash, "A Survey on Applications of Wireless Sensor Network Using Cloud Computing," *International Journal of Computer Science & Emerging Technologies (E-ISSN: 2044-6004)*, Volume 1, Issue 4, December 2010.

[2] W. Jiang, *Introduction and overview of Wireless Sensor Networks*. IGI Global, 2010.

[3] E. Ayars, "Using Xbee transducers for wireless data collection," *Apparatus and Demonstration Notes*, vol. July 2010, no. Am. J. Phys., Vol. 78, No. 7, 19 april 1010.

[4] N.S.A. Zulkifli, "XBee Wireless Sensor Networks for Heart Rate Monitoring in Sport Training".

[5] M. Bohmer, *Beginning Android ADK with Arduino*. Germany: Apress, 2012.

[6] G. Brandt, "Efficient data collectoin using Android ADK in a high velocity, mobile environment," May 1, 2012.

[7] B. Kaufmann, "Computing, Design and Implementation of a Toolkit for the Rapid Prototyping of Mobile Ubiquitous," August 2010.

[8] A. Allan, *Ios Sensor Apps With Arduino, Wiring The Iphone And Ipad Into The Internet Of Things*. USA: O'Reilly, september 2011.

[9] X. S. Ya Liu, "Symposium on Electrical & Electronics Engineering," in *A Design of Simple Serial Communication Protocol Based on the*, China, 2012.

[10] M. Weldin, *Arduino Cookbook*. USA: O'Reilly, december 2011.

[11] H. Timmis, *Practical Arduino Engineering*. Apress, november 2011.

[12] J. Sarik, "Arduino prototyping platform," in *Lab kits using the Arduino prototyping platform*, Washington, Oct. 2010, pp. T3C-1 -T3C-5.

[13] M. Rijnbout, "SmartGoals: a Hybrid Human-Agent Soccer Training System," 2009.

[14] B Perumal, "WSN INTEGRATED CLOUD FOR AUTOMATED TELEMEDICINE (ATM) BASED e-HEALTHCARE," in *IPCBEE vol.29 (2012)*, Singapore, 2012, p. vol29.

[15] S. Monk, *30 Arduino Projects for the Evil Genius*. Europe: McGraw-Hill Education, september 2010.

[16] M. Margolis, *Robot, Make an Arduino-controlled*. USA: O'Reilly Media, november 2012.

[17] S. Mada, "An Adaptive embedded system for helping patients," *International Journal of Computer Trends and Technology*, p. volume2Issue2, 2011.

[18] T. Igoe, *Making Things Talk*. Canada: Dale Dougherty, O'Reilly, 2007.

[19] R. Faludi, *Building Wireless Sensor Networks*. USA: O'Reilly Media, januari 2011.

[20] C. Doukas, *Building Internet of Things with the Arduino*. Createspace, april 2012.

[21] M. Dijusto, *Environmental Monitoring with Arduino*. USA: O'Reilly Media, februari 2012.

[22] M. Banzi, *Getting Started with Arduino*. USA: O'Reilly Media, oktober 2011.

[23] D. Cox, "Time Synchronization for ZigBee Networks," *ZMD America, Inc.*, no. IEEE, 2005.

[24] R. Fry, *Processing. A Programming Handbook for Visual Designers and Artist*. United States of America: Massachusetts Institute of Technoogy, 2007.

Appendix A: Use case examples

Application 1: Dispena Soccer (FootballNote)

More and more, athletics Skills of players are becoming a relevant success factor in modern football. Football note has several exercises to teach athletic skills to young players. The video[10] shows how athletics skills can be measured in a laboratory. One of the most famous players, Christiano Ronaldo, has been used for this experiment.

Possible Use Cases:

Use case 1 - Measure the velocity of a player over a straight distance of 50 meters. It should include intermediate measurements after 5m, 15m, 25m, 40 meters. The unit of measure is meters per seconds.

Use Case 2 - Measure the velocity of a player on a slalom over a distance of 30 meters. It should include intermediate measurements after each bend in the slalom, as a measurement point. The unit of measure is meters per seconds (figure 25).

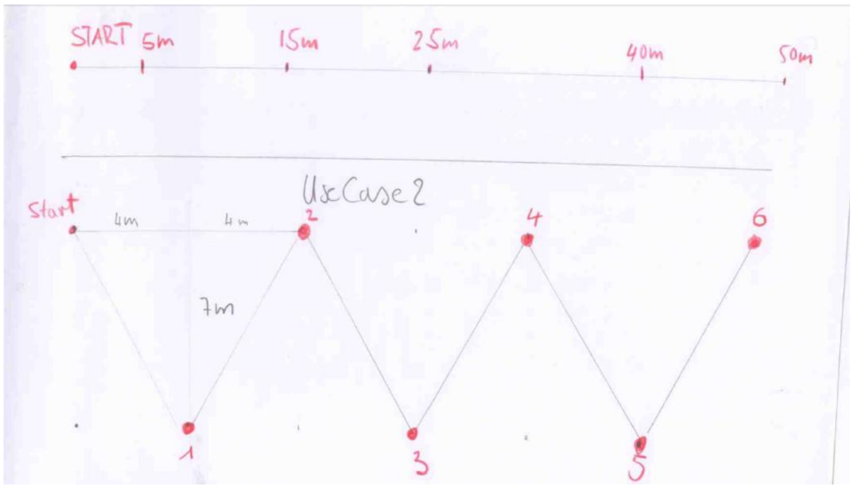


Figure 25: Use Case 2

[10] http://www.youtube.com/watch?v=VpMhf5XWNEw

Use Case 3 - Measure the jumping force (Newton) and jumping height (m) of a player (figure 26)

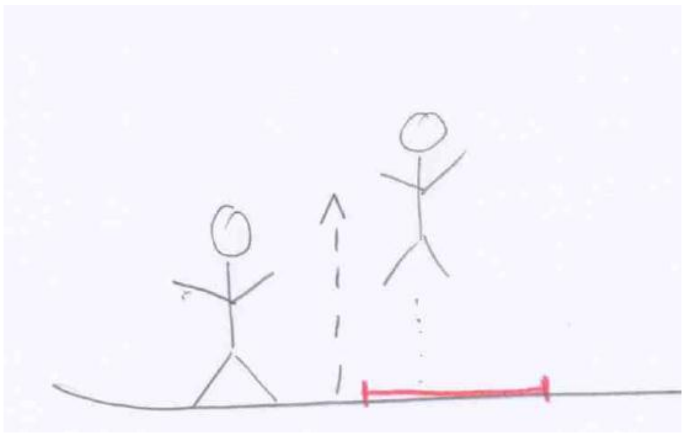


Figure 26: Use Case 3

Training Case 4 - Endurance of a player. Measure the heart rate while the player is running a circuit of 150m (6 times). The circuit is a quadrant with four measurements point. At every measurement point, the time and heart rate should be recorded (figure 27).

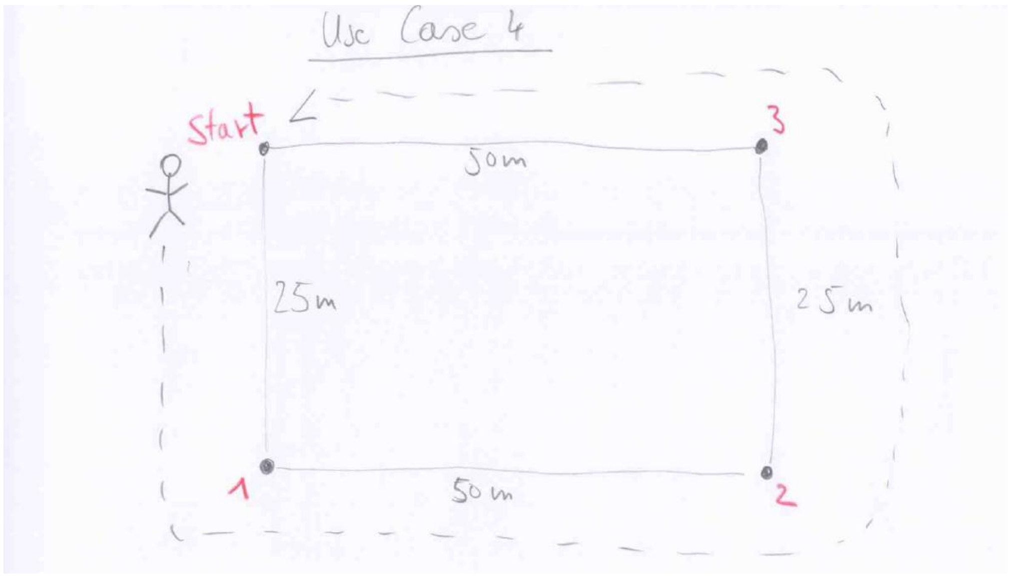


Figure 27: Use Case 4

Training Case 5 - Measure the movements of a player. Based on the similar approach of a Nike Sensor (see: http://www.youtube.com/watch?v=CcMDyPyHMUc), it should be possible to measure the step frequency, actual position in a quadrant and the distance run using the Arduino. (figure 28)

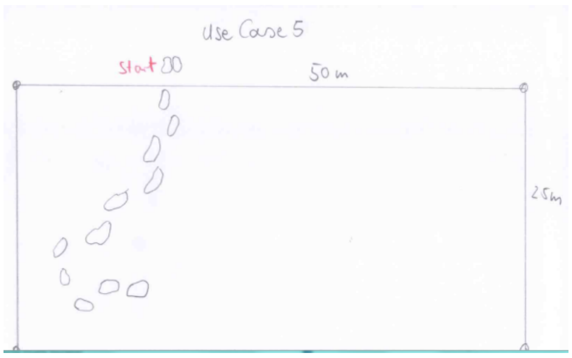


Figure 28: Use Case 5

Training Case 6 - Shooting Power. – With an HF Sensor[11], it should be possible to measure the shooting power of a player. It would be enough to save the speed through Arduino in an Oracle Database, and also recording the time of shoot.

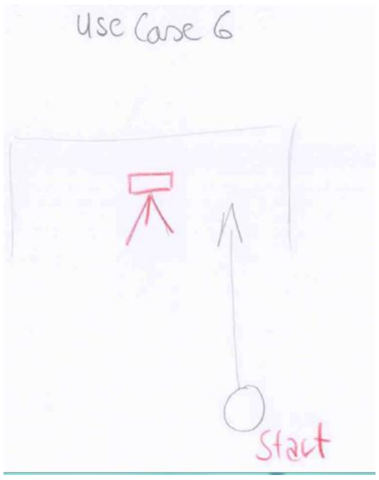


Figure 29: Use Case 6

[11] http://www.sportco.de/teamsport/fussball/trainingshilfen/10477/speed-radar-control?c=186751
And test: http://www.youtube.com/watch?v=qcxriNLAx10&feature=related.

Use Case 7 – Logistic - GPS (before reaching the store). Firstly an object will be recognized by an RFID label. After recognizing the following properties of the object will be measured: temperature, weight, size, humidity.

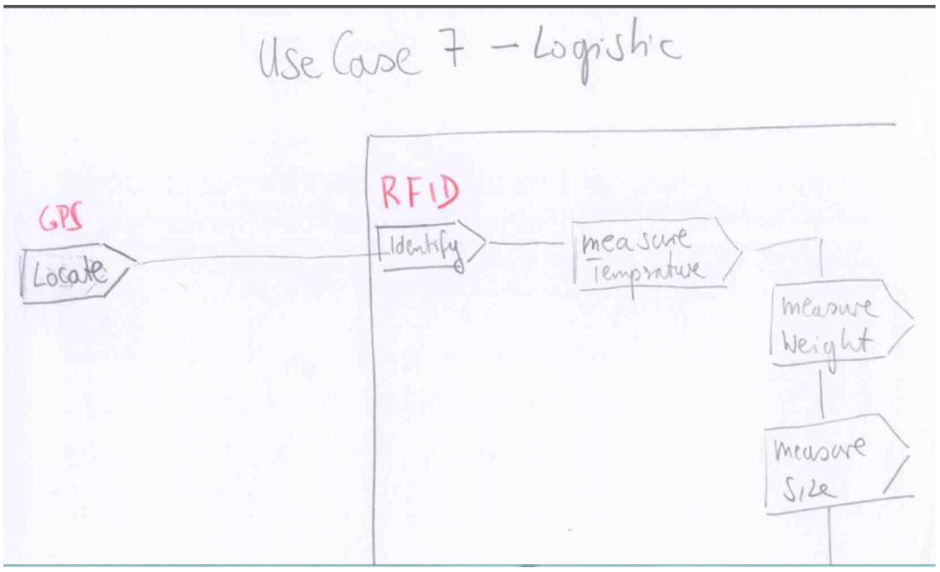
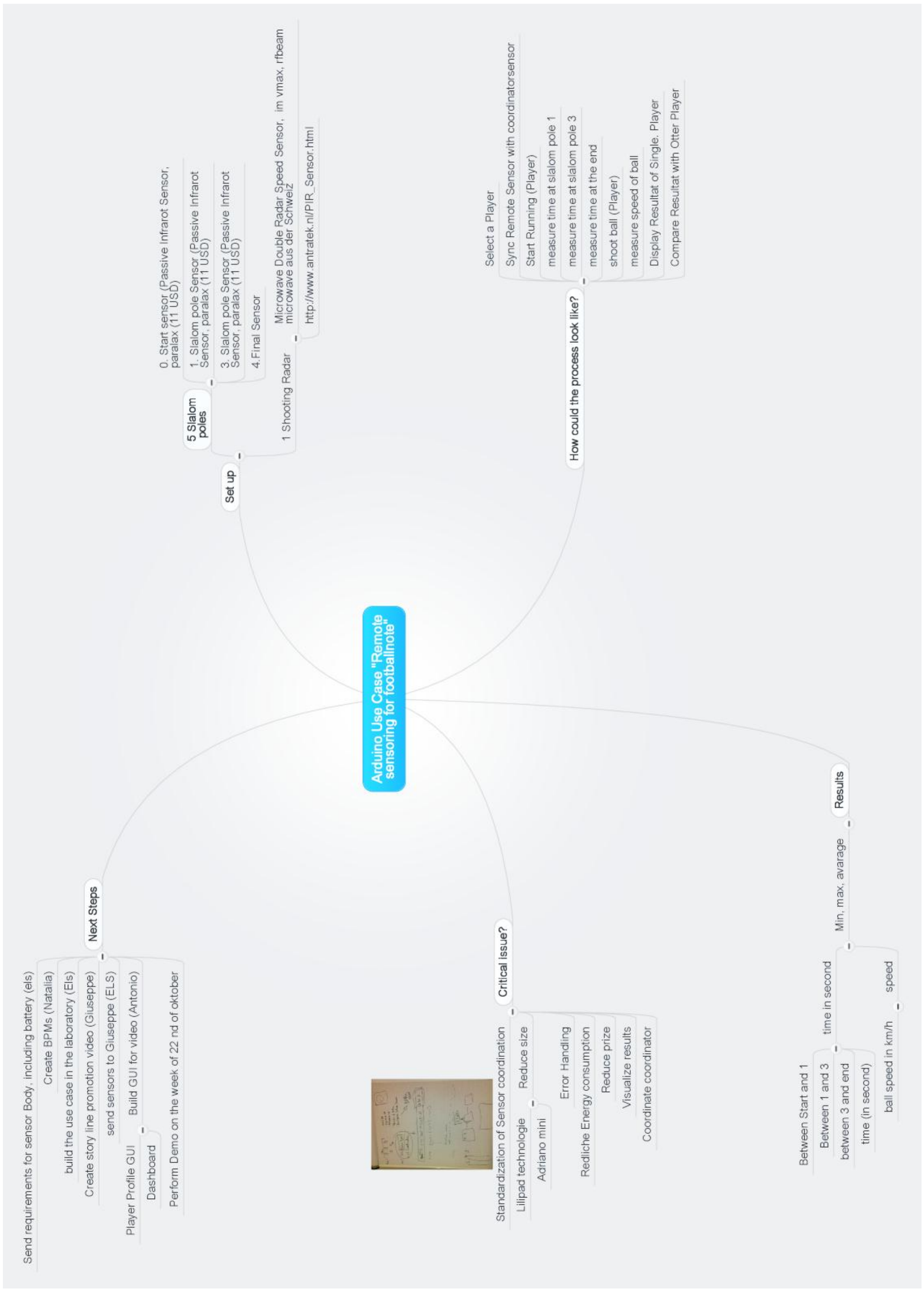


Figure 30: Use Case 7

Use case 8 - Manufacturing (Quality Insurance)

The manufacturing experiment should be done with a more robust sensor, which is able to measure in more difficult environments and for e.g. air pollution, identifying a fracture, etc.

Summary of initial project meetings



Arduino Use Case "Remote sensoring for footballnote"
Set up
5 Slalom poles
0. Start sensor (Passive Infrarot Sensor, paralax (11 USD)
1. Slalom pole Sensor (Passive Infrarot Sensor, paralax (11 USD)
3. Slalom pole Sensor (Passive Infrarot Sensor, paralax (11 USD)
4.Final Sensor
1 Shooting Radar
Microwave Double Radar Speed Sensor, im vmax, rfbeam microwave aus der Schweiz
http://www.antratek.nl/PIR_Sensor.html
How could the process look like?
Select a Player
Sync Remote Sensor with coordinatorsensor
Start Running (Player)
measure time at slalom pole 1
measure time at slalom pole 3
measure time at the end
shoot ball (Player)
measure speed of ball
Display Resultat of Single. Player
Compare Resultat with Otter Player
Next Steps
Send requirements for sensor Body, including battery (els)
Create BPMs (Natalia)
build the use case in the laboratory (Els)
Create story line promotion video (Giuseppe)
send sensors to Giuseppe (ELS)
Build GUI for video (Antonio)
Player Profile GUI
Dashboard
Perform Demo on the week of 22 nd of oktober
Critical issue?
Standardization of Sensor coordination
Lilipad technologie
Reduce size
Adriano mini
Error Handling
Rediche Energy consumption
Reduce prize
Visualize results
Coordinate coordinator
Results
Min, max, avarage
time in second
Between Start and 1
Between 1 and 3
between 3 and end
time (in second)
speed
ball speed in km/h

Figure 31: Mindmap

Appendix B: Arduino

An Arduino is a small microcontroller board. There are many different Arduino boards; the main differences between them are costs, type of processor, processor speed, number of input/output ports, etc. The standard Arduino is called the Arduino Uno. This project uses the Arduino Uno.

The Arduino Uno (figure 32) is a small microcontroller board based on the *ATmega328 microprocessor*. This means:

It has 14 digital input/output pins. Another Six pins can be used for Pulse-With-Modulation (PWM), e.g. necessary for controlling an DC-motor. The Arduino Uno is also fitted with a 16 *MHz* crystal oscillator that is used for timing (more specifically: giving the µC its clock signal).

A USB plug is needed to connect the Arduino to the computer. There are three ways to power this board, with a USB connection (from the computer or other USB source), using a battery or using a normal external power supply.

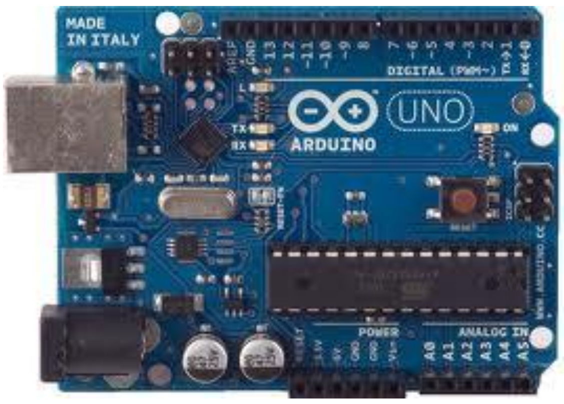
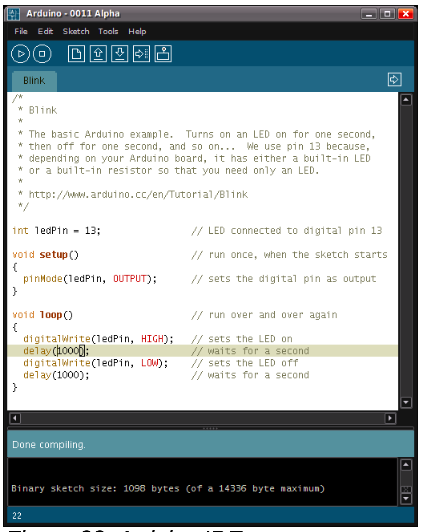


Figure 32: Arduino



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

```
Done compiling.

Binary sketch size: 1098 bytes (of a 14336 byte maximum)
22
```

Figure 33: Arduino IDE

The Arduino IDE (figure 33), which is a development environment and compiler, is used for creating the programs and can be used for different Arduino types.

Programs for the Arduino are called sketches. The sketches are programmed in a special developed embedded C variant.

The sketches can be uploaded to the Arduino board, when connected to the PC using a USB cable.

The block diagram of the Arduino (see figure 34) demonstrates the architecture of the board.

The heart of the device is the Central Processing Unit (CPU). It fetches program instructions stored in the Flash memory and executes those. The Arduino has a small working memory (32kb). An EEPROM (non-volatile storage) is used for some basic functionality (in order to turn the board on and off, etc.). The card also had some input / output for communication and sensing or controlling external electronics devices connected to the ports.

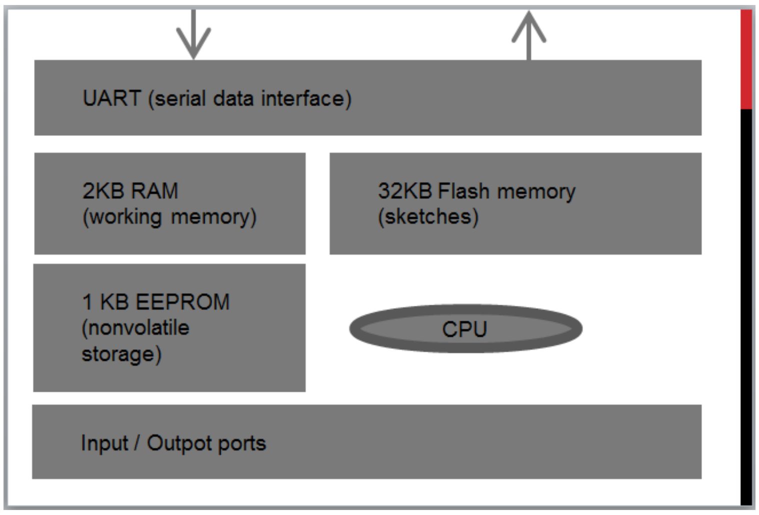
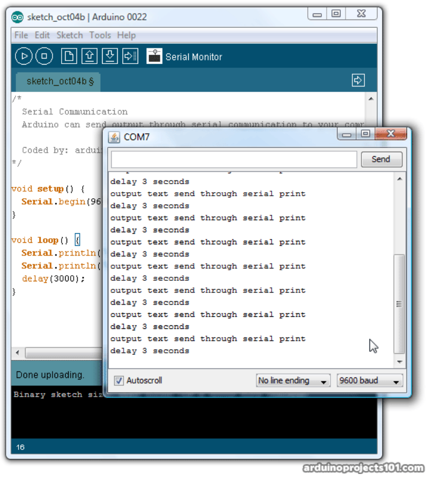

UART (serial data interface)
2KB RAM (working memory)
32KB Flash memory (sketches)
1 KB EEPROM (nonvolatile storage)
CPU
Input / Outpot ports

Figure 34: Diagram of the Arduino



```
/*
  Serial Communication
  Arduino can send output through serial communication to your comp

  Coded by: ardui
*/

void setup() {
  Serial.begin(96
}

void loop() {
  Serial.println(
  Serial.println(
  delay(3000);
}
```

```
delay 3 seconds
output text send through serial print
delay 3 seconds
output text send through serial print
delay 3 seconds
output text send through serial print
delay 3 seconds
output text send through serial print
delay 3 seconds
output text send through serial print
delay 3 seconds
output text send through serial print
delay 3 seconds
output text send through serial print
delay 3 seconds
```

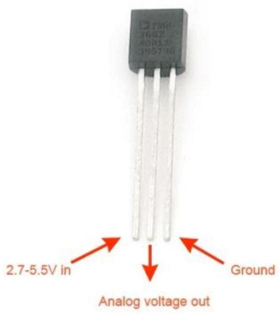
```
Done uploading.
Binary sketch siz
16
```

Figure 35: Serial monitor

All Arduino boards have at least one serial port (also known as a UART or USART): **Serial** (see figure 35). It communicates on digital pins 0 (RX [receiving]) and 1 (TX [transmitting]) as well as with the computer via USB, by using a special onboard communication chip. The Arduino IDE provides a serial monitor (Figure) to display serial data received by the Arduino, when the Arduino is connected to a PC using this Serial interface.

Sensing temperature sensor with an Arduino

Suppose one would like to read a temperature value. In this case, the temperature sensor must be connected to the Analogue input. A well-known temperature sensor is the TMP36, which must be connected to an Analogue input pin on the Arduino.

	TMP36: Temperature sensor. This analog temperature sensor is a chip, that exactly tells the ambient temperature. In case the sensor is connected to 5V, the following formula can be used to convert the 10-bit analog reading into a temperature: **Voltage in milliVolts = (*reading from ADC*) * (5000/1024)** This formula converts the number 0-1023 from the ADC into 0 - 5,000 mV. Then, to convert millivolts into temperature: **Temperature = (Voltage in milliVolts – 500) / 10 ^{o}C**

Steps to be undertaken for sensing temperature on the computer:

1. Build the electronic circuit.
2. Connect the Arduino using a USB cable to the computer.
3. Create a sketch (code 1).
4. Upload the sketch to the Arduino.
5. Open a serial monitor for watching the values.

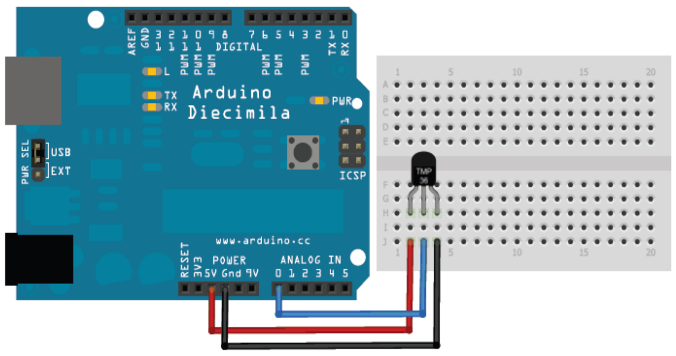


Figure 36: Temperature sensor with the Arduino

The Arduino sketch is straightforward, explanation of the code is given in the next table.

```
 int sensorPin = 0;
 void setup()  // this function runs once when you turn your Arduino on
 {
    Serial.begin(9600); //Start the serial connection with the computer
 }

 float readTemperature()  // reads the temperature in C
{
     //getting the voltage reading from the temperature sensor
     int reading = analogRead(sensorPin);

     // converting that reading to voltage, for 3.3v Arduino use 3.3
     float voltage = reading * 5.0;
     voltage /= 1024.0;

     // print out the voltage
     Serial.print(voltage); Serial.println(" volts");

   //converting from 10 mv per degree with 500 mV offset
    float temperatureC = (voltage - 0.5) * 100 ;
   return temperatureC;
 }

  void loop()  // run over and over again
  {
     float temperatureC = readTemperature();
     int integerPart  = (int)  temperatureC;
     int fractionPart = (int)  (temperatureC – integerPart) * 10);

     //  print the temperature
    Serial.print (integerPart); Serial.print(“.”); Serial.print (fractionPart); Serial.println ("oC");
    delay(1000); //waiting a second
  }
```

When opening the Serial monitor. The temperature will be printed out every second, e.g. like:

```
20.5 oC
20.5 oC
20.8oC
…
```

The Arduino uses a Serial interface, based on Asynchronous Serial Communication. It is the standard for communication between the Arduino board and a computer or other devices (**Fout! Verwijzingsbron niet gevonden.**). When using the Serial pins, either the pins of the Arduino board (RX / TX pins) are connected to another device, or a USB cable is plugged in.

In this sensor example the Arduino is connected to a personal computer through a USB port. This involves the Universal Serial Bus protocol.

RX/TX: it involves receiving and sending data through the so called TTL[12] protocol, and is often used when an additional boards (called: shield) is attached to the Arduino. It means that the RX (RX is the abbreviation of receiver) from the sender should be connected to the TX (TX is the abbreviation of transaction) of the receiver, and the other way around (figure 37).

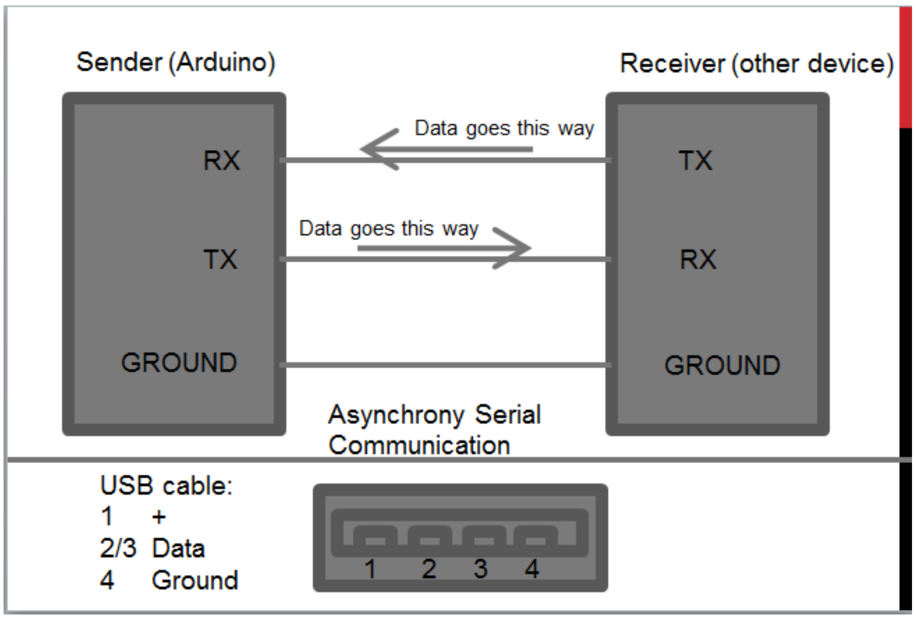


Figure 37: Asynchrony Serial Communication

These two connection types involve the use of another data protocol. Table 20 gives an overview of the meaning of these protocols in the OSI (***Open Systems Interconnection)*** layers.

[12] TTL, or in order words: transistor-transistor logic (TTL) voltage levels have been 5.0 Volt, with a high being any voltage above about 3.5 Volt and a low being any voltage below about 1.5 Volt.

Table 20: TTL and USB protocol

Layer	TTL	USB protocol
Physical layer	Receiving and sending data over pin 0 and 1.	Sending data over two data lines (2 and 3). Line 2 carries the signal data on negative Voltage. And on line 3 as positive Voltage. Together it always adds up to zero (checksum).
Electrical layer	0 / 5 Volt	-5 / 0 / +5 Volt
Logical layer	5 volt signal = value 1 0 volt signal = value 0	-5 (line2) and – 5 (line3) = value 1 0 volt signal = value 0
Data layer	Data is sent at 9,600 bits per se-cond.	Data could be sent up to 480 meg-abits. The maximum the Arduino can handle is: 115,200 bits per second.

These two protocols are adapted in the Serial Interface library from the Arduino (see Appendix B).

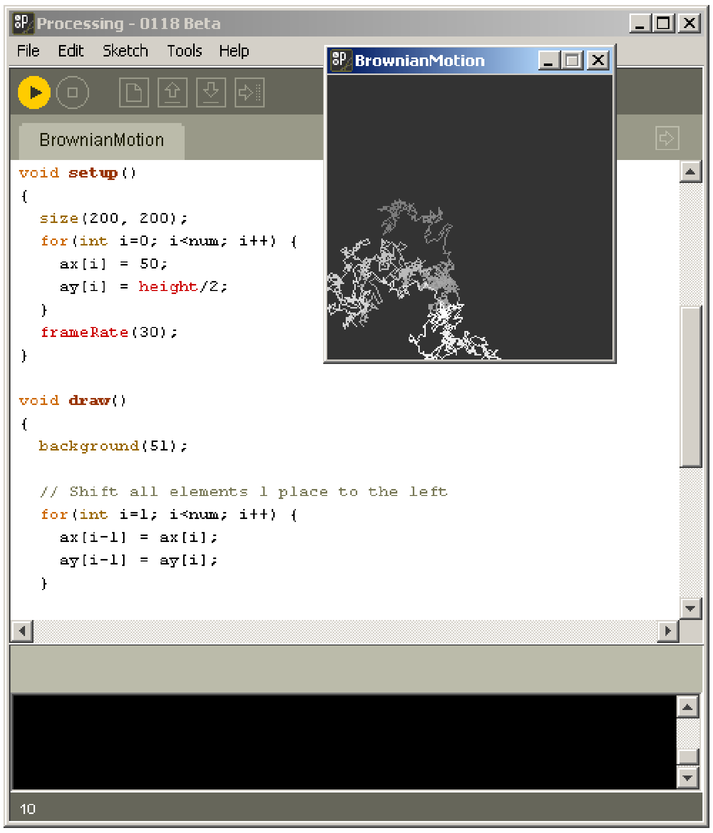
Of course, other protocols exist for Wireless and Wired Communication techniques. Table 21 shows the different technologies, including their data communication rate and the operation range.

Table 21: Serial Communication

Command	***Description***
begin()	Sets the data rate in bits per second (baud) for serial data transmission. Depending on the communication module, the following rates can be used: 300, 1,200, 2,400, 4,800, 9,600, 14,400, 19,200, 28,800, 38,400, 57,600, or 115,200 bits per second.
end()	Disables serial communication, allowing the RX and TX pins to be used for general input and output.
available()	Get the number of bytes (characters) available for reading from the serial port.
read()	Reads incoming serial data.
peek()	Returns the next byte (character) of incoming serial data without removing it from the internal serial buffer.
flush()	Waits for the transmission of outgoing serial data to complete.
print()	Prints data to the serial port as human-readable ASCII text.
println()	Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'
write()	Writes binary data to the serial port. This data is sent as a byte or series of bytes.
SerialEvent()	Called when data is available. Use Serial.read() to capture this data.

Appendix C: Processing

Processing (Figure 38) is an open source programming language and environment for people who want to create images, animations, and hardware interactions. It is specially designed for smooth communication with the Arduino. The code itself is similar to the Arduino sketches, but the language is more extendable as will be explained next.



```
void setup()
{
  size(200, 200);
  for(int i=0; i<num; i++) {
    ax[i] = 50;
    ay[i] = height/2;
  }
  frameRate(30);
}

void draw()
{
  background(51);

  // Shift all elements 1 place to the left
  for(int i=1; i<num; i++) {
    ax[i-1] = ax[i];
    ay[i-1] = ay[i];
  }
```

Figure 38: Processing

Since it is possible to compile and/or export the code as: Android, Windows, Linux, Mac application, it is platform independent. It is a great advantage to have the same programming and Graphical User Interface running on different devices.

Both programs (Arduino IDE and Processing IDE) use the **startup()** function to perform initialization procedures. Processing uses the **draw()** function for continuous output, while in Arduino the ***loop()*** is used.

The Processing language uses almost the same library for Serial communication.

The strongest point of using the Processing language of the Graphical User Interface is that it can run on different devices (see table 2).

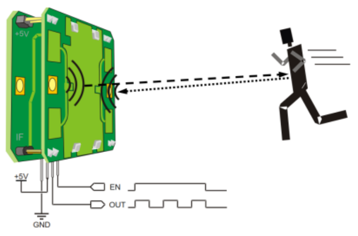
Table 22: Support of the Processing language

Support	Android Mobile	iPhone	iMAC	Linux	PC	JavaScript	Java Applet
Processing	🟩	🟥	🟩	🟩	🟩	🟩	🟩
Processing connected with an Arduino-ADK board	🟩	🟥	🟩	🟩	🟩	🟥	🟩

Appendix D: Development of a new shooting-power sensor

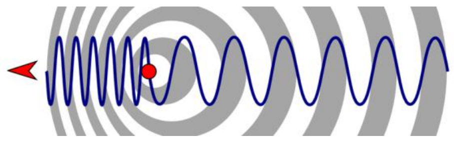
The X-band sensor is basically a radar detector.

How it works



The sensor transmits a high frequency radio wave and the reflection of the moving object is received.

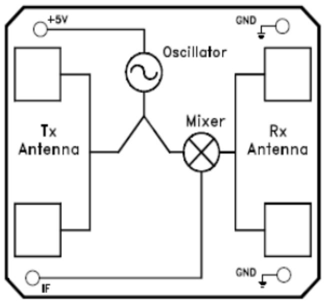
If the object moves, the received frequency will be different from the send frequency because of the Doppler effect.

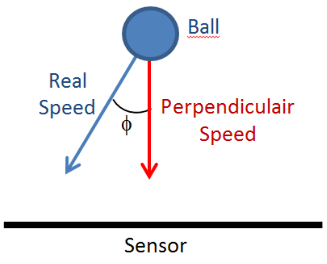


The Doppler effect will change the frequency of the reflected wave.

The relative changes in frequency can be explained as follows: When the source of the waves is moving toward the observer, each successive wave crest is emitted from a position closer to the observer than the previous wave. Therefore, each wave takes slightly less time to reach the observer than the previous wave. Therefore, the time between the arrival of successive wave crests at the observer is reduced, causing an increase in the frequency.

The frequency of the returned wave will be $f_r = \left(1 + 2\frac{v}{c}\right) f_0$, where *v* is the speed of the object (the ball) perpendicular to the emitted wave, *c* is the speed of light and f_0 the original frequency of the transmitted wave.





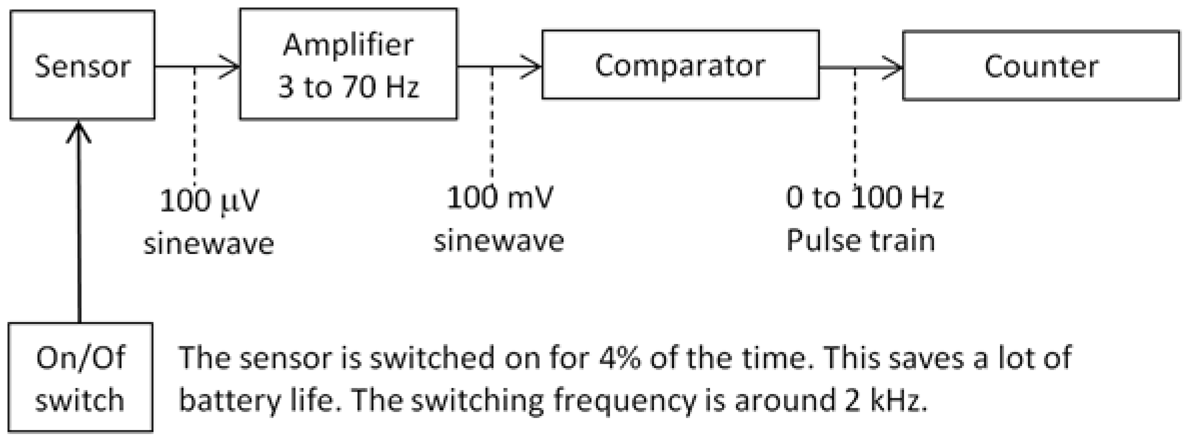
To detect the frequency change, the received signal is mixed (multiplied) with the original signal and available on a pin on the outside of the sensor (IF).
This multiplication of two signals with different frequencies will lead to two new signals, one with the sum of frequencies and the other with the differences of the two frequencies.
The Difference signal has a frequency in the audio range and is:

$$f_\Delta = \left(1 + 2\frac{v}{c}\right) f_0 - f_0 = 2\frac{f_0}{c}\ v \approx 18.5\ Hz\ km^{-1}h$$

The Sum signal is of no value. Because of its very high frequency it is (conveniently) lost through stray capacitances.
Unfortunately, the ball will normally not be moving perfectly perpendicular.
As a result, the resulting frequency of the output signal will be:
$f_\Delta = \frac{f_0}{c}\ v\ \cos\phi$, which means that we will underestimate the real speed.
We have to make sure that the players shoot from enough distance and straight on the goal.
Alternatively we can have multiple sensors on different frequencies and use that to calculate the real speed and direction of the ball.

The electronics in the Parallax module

Readymade modules are intended for motion detection and not for speed measurement. The electronics are always very similar. For the Parallax sensor it looks like:


Sensor
Amplifier 3 to 70 Hz
Comparator
Counter
100 µV sinewave
100 mV sinewave
0 to 100 Hz Pulse train
On/Of switch
The sensor is switched on for 4% of the time. This saves a lot of battery life. The switching frequency is around 2 kHz.

The limited bandwidth of the amplifier makes the design simpler in three ways:

1. The small bandwidth limits noise and disturbance, which makes the design of the amplifier simpler.
2. The sensor is switched at about 2 kHz, this saves battery power. That 2 kHz modulation needs to be suppressed in the output signal and choosing it much lower than the switching frequency makes suppression easier.
3. The resulting pulse train is also of low frequency and can be easily handled by a Arduino microcontroller, motion detection can be done by counting pulses, which is simply to implement and robust.

This schematic layout is optimized for motion detection but unusable for measuring speed. The sensor output is about 18 Hz km^{-1}.h and with a limit of 70 Hz it means that objects moving faster than 5 km.h^{-1} are not detected.

How to change the electronics and software

First of all the sensor need to operate in continues mode. This will avoid the 2 kHz modulation that needs to be suppressed in the output signal.

The amplifier then needs to be replaced by a low-noise version with a bandwidth of 100 – 2,000 Hz. For speed measurement, we are not interested in low frequency signals, or objects moving slower than 5 km.h^{-1}, starting from 100 Hz also supresses line voltage interference (at 50 Hz).

The resulting signal is than digitized by a comparator and fed into the microcontroller. Now we cannot simply count pulses, but we need to measure and process the time difference between consecutive pulse edges. This can be done using the capture unit on the 16 bit timer in the ATmega microcontroller. Care must be taken to handle timer overflow and making sure the Arduino has enough resources to handle all timer interrupts.

With careful amplifier design and software construction it is possible to measure speeds of more than 5 km.h^{-1}.