Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Transmission, storage, and visualization of data with ANT+

Examensarbete utfört i Informationskodning
vid Tekniska högskolan vid Linköpings universitet
av

Marcus Ericsson

LiTH-ISY-EX--15/4874--SE

Linköping 2015

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Transmission, storage, and visualization of data with ANT+

Examensarbete utfört i Informationskodning
vid Tekniska högskolan vid Linköpings universitet
av

Marcus Ericsson

LiTH-ISY-EX--15/4874--SE

Handledare:	**Harald Nautsch**
	ISY, Linköpings universitet
	Åsa Detterfelt
	MindRoad AB
Examinator:	**Robert Forchheimer**
	ISY, Linköpings universitet

Linköping, 10 juni 2015

	Avdelning, Institution Division, Department Avdelningen för Informationskodning Department of Electrical Engineering SE-581 83 Linköping	**Datum** Date 2015-06-10

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ ________	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ ________	**ISBN** — **ISRN** LiTH-ISY-EX--15/4874--SE **Serietitel och serienummer** Title of series, numbering **ISSN** —

URL för elektronisk version

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-154874

Titel
Title

Överföring, lagring och visualisering av data med ANT+

Transmission, storage, and visualization of data with ANT+

Författare
Author

Marcus Ericsson

Sammanfattning
Abstract

Ultra låg-strömförbruknings sensorer används av apparater som förbrukar lite energi. ANT+ sensorer kan köras i åratal på ett knappcells batteri. I denna rapport så kommer data från ANT+ sensorer att användas i en applikation som ska spara och visa data.

Nyckelord
Keywords

ANT, ANT+, wireless sensors, ultra low-power sensors, Android

Sammanfattning

Ultra låg-strömförbruknings sensorer används av apparater som förbrukar lite energi. ANT+ sensorer kan köras i åratal på ett knappcells batteri. I denna rapport så kommer data från ANT+ sensorer att användas i en applikation som ska spara och visa data.

Abstract

Ultra low-powers wireless technology sensors uses when devices is used to consumes low power. ANT+ sensors can run for years on a single coin battery. In the thesis the ANT+ sensor data is used in an application that can store and visualize the data.

Acknowledgments

Jag vill tacka Anders och Åsa på MindRoad som har gett mig en en rolig och lärorik uppgift. Jag vill tacka Harald som varit min handledare och Robert som varit min examinator.

Linköping, Juni 2015
Marcus Ericsson

Contents

List of Figures

Notation

Abbreviations

Abbrevation	Meaning
ANT	Advanced Network Tools
ANT-FS	ANT File Share
FIT	The Flexible and Interoperable Data Transfer
API	Application Programming Interface
UI	User interface
ISM	The industrial, scientific and medical radio bands
RF	Radio frequency
OSI	Open Systems Interconnection model
IEEE 802.15	Family of IEEE standards dealing with local area networks and metropolitan area networks.
Mbps	Megabit per second
Kbps	Kilobit per second
SQL	Structured Query Language
CRC	Cyclic Redundancy Check
LSB	Least significant bit
MSB	Most significant bit
Byte	Eight bits
Bit	Represented as either a 0 or 1

1 Introduction

With ultra low-power wireless technology a whole new market opens up, based on needs for devices that consume low power. It can be industrial uses where low-power sensors are used, to the home environment when one wants to get data from a thermometer, window or door sensor, to controlling the lights. Another large segment of products is in the health and fitness industry. There are already a lot of different sensors on the market (Heart Rate sensors, speed and cadence sensors) to connect with a smartphone or a watch.

In this master thesis I'm going to explore how ANT+ devices can send data and visualize the data in both an Android application and a Windows environment, and how to store the data.

1.1 Thesis objective

The main objective for this thesis is to collect data from an ANT+ sensor to an Android application. The application shall be able to visualize the data on the screen of the device and save into a database.

1.2 Problem definition

Based on the main objective, the problem definition for this thesis are:

- How to connect and collect data from ANT+?
- How to send collected data from FIT-files with ANT-FS?
- How to visualize ANT+ data in an application?

1.3 Background

This Master thesis has been implemented at MindRoad AB. MindRoad is a consulting company that works with software development, education and sourcing with the headquarter in Linköping. They have chosen to explore and create a wider understanding about how ANT/ANT+ can be implemented and used, and to build a platform for future projects.

ANT is an ultra low-power wireless protocol that operates in the 2,4 GHz ISM-Radio band. ANT is owned by ANT Wireless a part of Dynastream Innovations Inc. ANT's main target is the sports and fitness industry. ANT+ is an interoperability protocol on top of the ANT protocol. The sport and fitness industry is growing, a lot of athletes today use many different smartphone applications to record and save data from their workout. Today's technologies with smartphones make it easier to be able to use their phone in their training. Even the major smartphone operating system manufacturers themselves have released their own applications to their systems. On June 2014 Apple showed their iOS8 with the application, Healthcare, an application that saves fitness and health data to the user [2]. On June 25, 2014 Google showed their new training and fitness application Google Fit in the release of the new Android 5.0. Google Fit is an application that stores the fitness and health data of the user [6]. This Christmas gift for the year by HUI Research is the fitness band. HUI Research justifies its choice with:

"The fitness band has been on the market for some time now but it is only this year the sales are gaining momentum. The fitness band records activity as a pedometer but it also contains sensors that register pulse and sleep. The data from the fitness band is transferred to a computer or a mobile device generating an overview on present activities or sleep. It can also recount for tendencies concerning longer periods of time. The strong trend in health makes more people aware of their health situation and the connection between activity and well-being. Another trend in society is sharing experiences and pictures in social medias. These trends in health, digital wearables and social media sharing coincide with the fitness band and makes it the perfect The Christmas Gift of the Year in 2014." [1]

2 Methodology

Figure 2.1 shows the overview of the work process in this thesis. The method for the projects workflow is made by Wenell and called the Wenell Project Workflow (WPW)[25]. The work is structured to use Scrum to work in a agile way in the implementation phase. WPW is devided into six different phases: Concept, Prestudy, Start, Realization, Closure and Utilization.

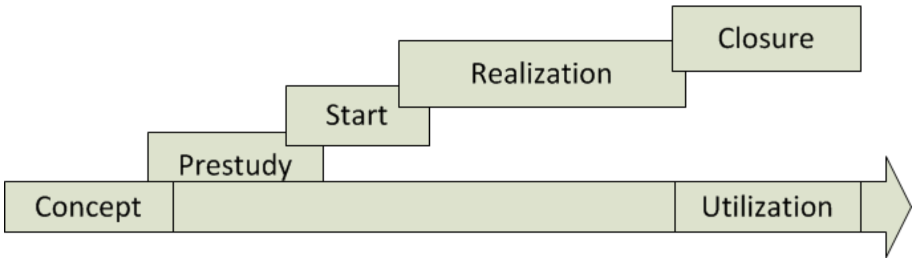


Figure 2.1: *Wenell Project Workflow*

2.1 Concept

In the concept phase the task was defined and the project was formed.

2.2 Prestudy

At the beginning of the thesis, a literature study was made to get more knowledge in the different areas that this thesis covers. About ANT and ANT+ the thisisANT.com site which is the official Web site of the ANT+. The site describes what ANT+ is and how it works. For the Android application part of the thesis, the information is taken from android developer site.

2.3 Start

In the start phase of the project the problem definition and limitations were defined. Also, the planing of the realization phase to visualize the project and organize what to do, and what not to do is done in this phase.

2.4 Realization

The realization phase consists of the system design and the implementation of the design. This phase will use the Scrum development method.

2.4.1 System design

The system design process is designed for the requirements of the system. No special method for the system design have been used, more than that an agile development design is used. The system design shows the prototype of the whole system. The system design is a mix of the requirements and a basic overview of the system's functionality. Most of the user stories of the product backlog used for Scrum is produced in the system design step.

2.4.2 Implementation

The implementation of the application will use Scrum, an agile software development method. The Android application are implemented using Android studio and the code is developed in the Java programming language with support of the Android API, ANT+ Android API and Garmin FIT API. Git is used for version control and backup.

2.4.3 Scrum

Scrum is an Agile software development method. Scrum is divided into the three core user roles: Product owner, Development team and Scrum master. The choice of scrum is because it is lightweight, simple to understand and a popular agile software development method. [16]

The Scrum process is divided into four steps, shown in figure 2.2:

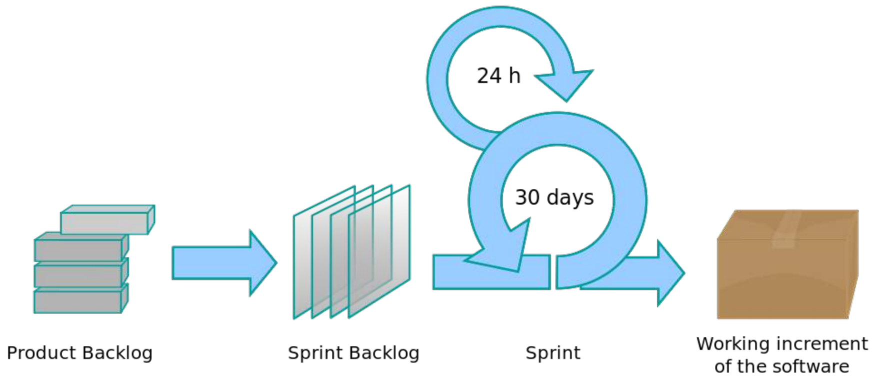


Figure 2.2: *The Scrum Process. [26]*

2.4.3.1 Product backlog

The product backlog is an ordered list of everything that might be needed in the product. The product backlog lists all the requirements, features and functionality of the product. The product backlog is dynamic, it can always be changed to identify the product needs.

2.4.3.2 Sprint backlog

Sprint backlog is a set of Product backlog items that is selected for each sprint. The sprint backlog is dynamic, when new work is required new tasks is added to the sprint backlog.

2.4.3.3 Sprint

At the beginning of a sprint, a sprint plan is made. The sprint planning answers the following: What can be done in the sprint? and How will the chosen work get done to reach to the sprint goal?

2.4.3.4 Working increment

After the sprint the product can be a deliverable product or there can be a new sprint to add new functionality of the user story's in the Product backlog.

2.5 Testing

For a system to work correctly and without errors it will need a certain amount of testing of the system. There are many things that can be accomplished with testing, everything from performance to check that the system behaves as planned. Overall the tests will be divided into two different categories:

- Unit testing
- UI testing

2.5.1 Unit testing

To test a function, smaller part or an feature in the application code, the unit testing is applied. A Unit test goal is to test a isolated part of the program to show that just that part is individually correctly tested. For the Java programming language, JUint is being used as a testing framework.

2.5.2 User Interface Testing

For testing the behavior of the User Interface (UI) the Android SDK comes with tools that support automated functional UI testing: [7]

- uiautomatorviewer - A GUI tool to scan and analyze the UI components of the application
- uiautomator - A Java libary thats contains a API to create customized functional UI tests

2.6 Closure

In the closure phase the results are presented, evaluated, documented and the product that has been developed is delivered. This phase also includes the finishing and closing of the project.

2.7 Evaluation

The evaluation of the product can be done in many different ways. The evaluation will be done both on the method that is chosen and how the Realization phase has been completed.

3
The ANT wireless protocol

3.1 ANT

ANT is a wireless sensor network running in the license-free 2,4GHz ISM band. It is designed for ultra-low power usage (ULP), to target battery operated devices that require many years of battery life (often a single coin cell). ANT is suitable to any kind of low data rate network typologies, such as mesh, star or peer-to-peer connection. Each ANT channel can be independently configured to either be a one-way or two-way communication channel. ANT is specifically designed for wireless sensor networks (WSN) that require [11]:

- ultra low power - runs on a coin cell for years of operation
- highly resource optimized - fits into a compact sized chip
- network flexibility and scalability - self-adaptive and able to do practical mesh
- easy to use with low system cost - operates independently with a single chip

ANT devices can use any Radio frequency (RF) range from 2400MHz to 2524MHz, with one exeption of 2457Mhz, which is reserved for ANT+ devices. ANT provides management of physical, data link, network and transport layers of the OSI stack shown in figure 3.1[15]

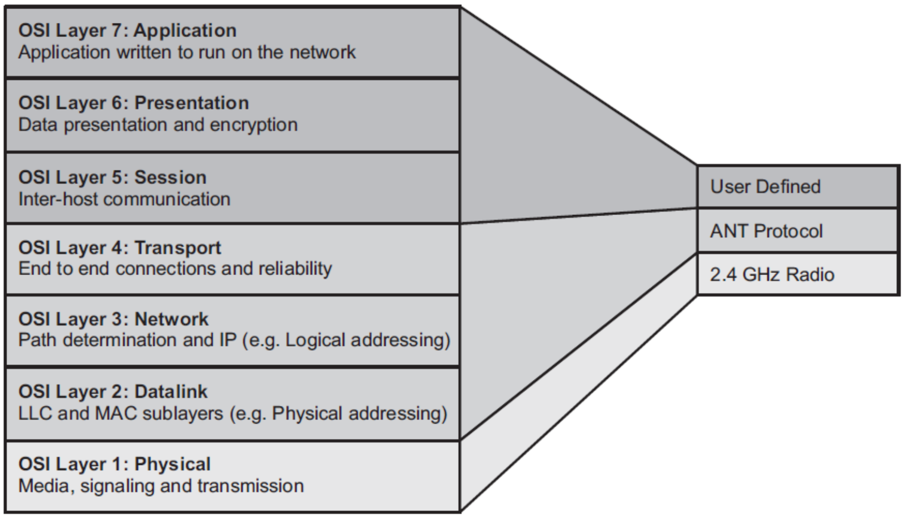

OSI Layer 7: Application
Application written to run on the network
OSI Layer 6: Presentation
Data presentation and encryption
OSI Layer 5: Session
Inter-host communication
OSI Layer 4: Transport
End to end connections and reliability
OSI Layer 3: Network
Path determination and IP (e.g. Logical addressing)
OSI Layer 2: Datalink
LLC and MAC sublayers (e.g. Physical addressing)
OSI Layer 1: Physical
Media, signaling and transmission
User Defined
ANT Protocol
2.4 GHz Radio

Figure 3.1: *ANT OSI Layer.*

3.2 ANT+

ANT+ is an interoperability protocol on top of the existing ANT protocol. It encourage interoperability and open-access data between manufactures of ANT+ devices. The target applications for ANT+ are wireless sensors for sport, wellness and home health. Examples of ANT+ devices are [11]:

- Heart rate monitor
- Bike speed and cadence sensors
- Bike power sensor
- Weight scale
- Fitness equipment data sensors
- Temperature sensor

All ANT+ devices have their own device profiles. Each profile determines the function of each device. For example the Environment profile is always sending out the same data independent of which manufacturer makes the device. All the devices are preconfigured so the ANT+ display units (PC, Smartphone, etc) always know what type of device/sensor they are getting data from [9] .

3.2.1 ANT+ Alliance

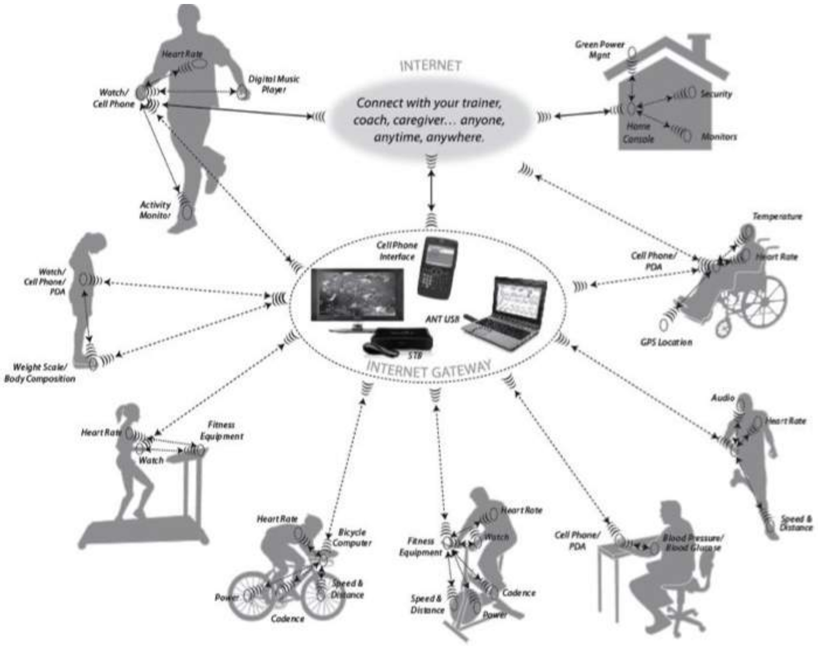


Figure 3.2: *ANT+ Device Ecosystem.*

The ANT+ Alliance is organized by Dynastream Innovations Inc, a subsidary of Garmin Ltd, the alliance has more than 350 members, including Adidas, Garmin, Suunto, Qualcomm, Samsung, Sony Mobile and Texas Instruments [14]. Figure 3.2 shows the ANT+ Device ecosystem, where the ANT+ is used to send data from sensors to the user.

3.3 ANT Network

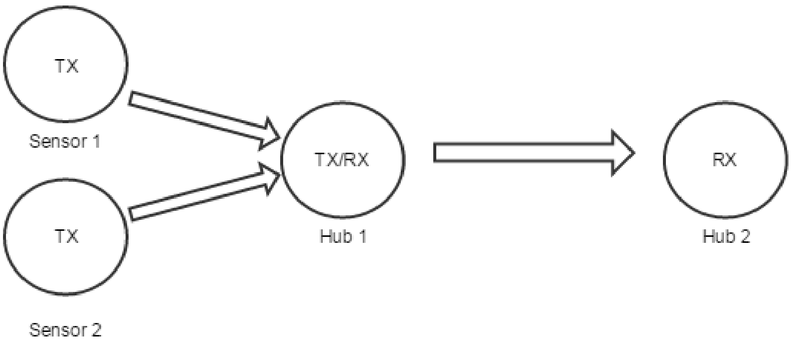


Figure 3.3: *Simple ANT Network.*

ANT usage and configuration is channel-based. Each ANT nod (represented by a circle) can connect to other nods via dedicated channels.

For communication between two nods a channel has to be established. A channel consists of:

- Master (ex. Sensor1).
- Slave (ex Hub1).

Figure 3.3 shows a simple ANT network consisting of, two sensors and two hubs. The masters act as the primary transmitters, and the slaves act as the primary receivers in this ANT network. This network is a typical ANT+ network. The Sensors can for example be an Environment sensor and a Heart rate sensor. The Hub1 could be a Smartphone that collects data from the sensors, Hub2 a PC that gets FIT-files from the Smartphone [13].

Channel	Master	Slave
Channel A	Sensor1 (TX-Only)	Hub1 (RX)
Channel B	Sensor2 (TX)	Hub1 (RX)
Channel C	Hub1(TX)	Hub2 (RX)

Table 3.1: *Simple ANT network.*

For the ANT+ channel a period can vary from 0.5Hz (1 message/every two second) to 4Hz (4 message/second).

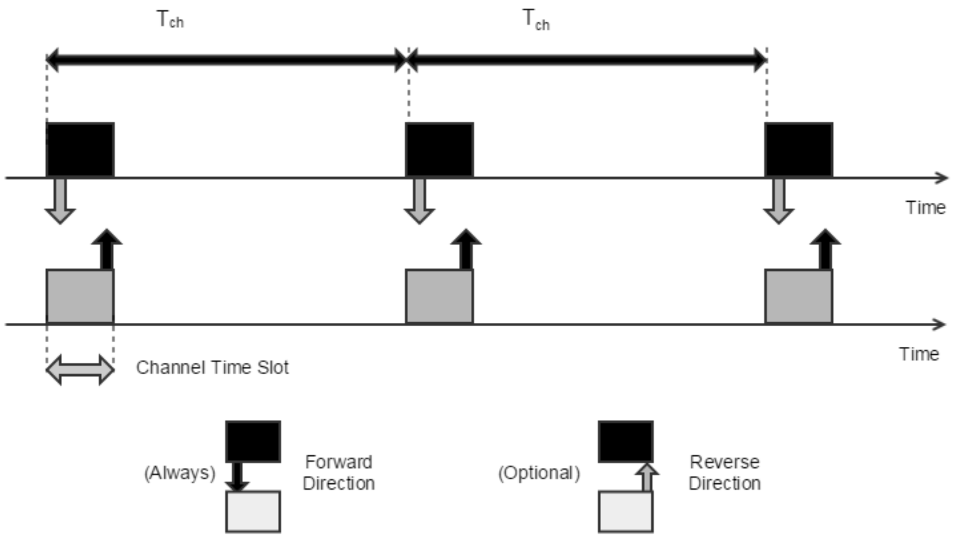


Figure 3.4: *ANT Channel Timing.*

The majority of ANT implementation of channels are synchronous, independent and bidirectional. When opening a synchronous channel, the master node always opens a search window to check that it transmission does not interface with transmissions of other devices. When the channel is open the master will always transmit a message on each channel time slot for it channel period (Tch) as shown in figure 3.4 [13].

Data messages are transferred between nodes in one of two directions[13]:

- Forward Direction (Master -> Slave)
- Reverse Direction (Slave -> Master)

3.3.1 ANT Data Types

The data types used for ANT is: broadcast, acknowledged, burst and advanced burst message transfers. Each data type is sent in 8 byte packets. [13]

Data Type	**Channel Direction**	**Description**
Broadcast	Forward	Default Data Type. Broadcast messages sending every timeslot and retransmitting if ANT does not receives any new data from the master.
	Reverse	Broadcast message optionally sent each timeslot, only if requested by the slave. Only sent once, no retransmission.
Acknowledged	Forward	Sent on the next timeslot, if requested.
	Reverse	Only sent if requested by slave. Cannot be send if no message is received. Only sent once, no retransmission.
Burst	Forward	A burst transfer will commence at start of the next timeslot. Burst packets synchronize each other. The last burst package will retransmit on the next channel period if no new data has been received.
	Reverse	Only sent if requested by slave. Cannot be send if no message is received. Only sent once, no retransmission.
Advanced Burst	Forward	An Advanced burst transfer will commence at start of the next timeslot. Burst packets synchronize each other. The last burst package will retransmit on the next channel period if no new data has been received.
	Reverse	Only sent if requested by slave. Cannot be send if no message is received. Only sent once, no retransmission.

Table 3.2: *ANT Data Types*

3.4 Channel Configuration

In order for two ANT devices to communicate, they require a channel configuration. ANT+ has it own channel configuration.

3.4.1 Channel type

There are different types of ANT channels, the channel type can be chosen depending on the application requirements, power consumption, topology, and battery life expectancy [13].

Value	Description
0x00	Bidirectional Slave Channel
0x10	Bidirectional Master Channel
0x20	Shared Bidirectional Slave Channel
0x30	Shared Bidirectional Master Channel
0x40	Slave Receive Only Channel (diagnostic)
0x50	Master Transmit Only Channel (legacy)

Table 3.3: *ANT channel Types*

The bidirectional channels support two-way communication. The shared channels can be used when a single node must receive from many nodes. Transmit/Receive channels can only send data from the master to the slave node (forward direction).

3.4.2 RF Frequency

ANT supports 125 unique radio frequencies (RF), from 2400Mhz to 2524Mhz. The default RF for ANT is 2466 Mhz. Some of the RF channels are regulated by the ANT+ Alliance, ANT+ is using 2450 MHz and 2457 MHz and should be avoided by non-ANT+ devices. The RF frequency is controlled by a 8-bit field with values ranging from 0 to 124. Equation 3.1 can be used to determine the value of the RF frequency.

$$RF\_Frequency\_val = \frac{Desired\_RF\_Frequency\_val - 2400(MHz)}{1MHz} \tag{3.1}$$

3.4.3 Channel ID

Channel ID contains Transmission Type, Device Type and Device Number.

3.4.3.1 Transmission Type

Transmission Type tells what type of channel that is being used, Reserved, Independent channel or Shared channel. A master device needs to define a transmission type.

3.4.3.2 Device Type

The device type determines the type of device, for ANT+ all device types are predefined by default. For example all the Heart Rate Sensors have the device type 120, independent of the manufactures. So all ANT+ displays (Smartphone or PC, etc) know what type of sensor they are getting data from.

3.4.3.3 Device Number

The device number is unique for any given device type. It can be a random generated number or a serial number.

3.4.4 Channel period

The channel period represents the message rate of the data packets. The master will send a packet on every time slot at the rate. The channel period ranges from 0.5Hz to 200Hz. The channel period is defined by 16-bit field with values ranging from 0 to 80. Equation 3.2 can be used to determine the channel period.

$$Channel\_Period\_val = \frac{32768}{MessageRate(Hz)} \tag{3.2}$$

To have a message rate of 4Hz the channel period must be set to 32768 / 4 = 8192. The default message rate is 4Hz. Proper assignment of channel period is critical and it is imperative to be mindful of the following issues:

- The message rate is proportional to the power consumption
- A small channel period allows higher data-transfer rates
- A small channel period results in faster successful device-search operations

3.4.5 Network

ANT supports public, managed and private networks. In order for two ANT devices to communicate, they need to be connected to the same network. ANT+ is a network that's managed by ANT+ Alliance.

3.4.6 Network Key

To access an ANT+ network, you need the network key. The network key for ANT+ is given out to ANT+ Adopters on the ANT website. The ANT+ network key may only be used by ANT+ devices.

3.4.7 Channel Configuration ANT+

Example of a channel configuration for the ANT+ slave device is:

Parameter	**Value**	**Description**
Network Key	ANT+ Network Key	Default ANT+ Network Key
RF Frequency	57	ANT+ frequency 2457MHz
Device Number	0	Wildcard (searching for all)
Transmission Type	0	Paring
Device Type	25	Environment Sensor
Channel Type	0x000	Slave Channel
Channel Period	65535	2Hz Message Rate

Table 3.4: *An example of a channel configuration for ANT+*

3.5 Message structure

3.5.1 Standard message format

The ANT serial message structure uses different length depending of what type of message is being sent. A typical message that begins with a SYNC byte and ends with a CHECKSUM has the following format:

Sync	Msg Length	Msg ID	Payload (8 bytes)	Checksum

Table 3.5: *Standard data message format*

3.5.2 Extended message format

In order to send more information like Channel ID(Device number, Device Type and Transmission type) as ANT+ use, extended data will be added to the data message. The Flagged extended data message format looks like table 3.6:

Sync	Msg Length	Msg ID	Channel number	Payload (8 bytes)	Flag byte	Channel ID	Checksum

Table 3.6: *ANT flagged extended data message format*

The ANT+ data message packet format shown in table 3.7:

Byte	Name	Length	Description
0	SYNC	1 Byte	Fixed value of 10100100 or 10100101
1	MSG LENGTH	1 Byte	Number of data bytes in the message
2	MSG ID	1 Byte	Data Type Identifier 0: Invalid 1..255: Valid Data Type
3	CHANNEL NUMBER	1 Byte	Channel number
4-11	PAYLOAD	8 Bytes	ANT+ payload data
12	FLAG BYTE	1 Byte	0x80 to indicate that channel ID information is present in the extended data bytes
13-14	DEVICE NUMBER	2 Bytes	ANT+ Profile device number
15	DEVICE TYPE	1 Byte	ANT+ Profile Device type
16	TRANS TYPE	1 Byte	ANT+ Profile Tranmision type
17	CHECKSUM	1 Byte	XOR of all previous bytes including the SYNC byte

Table 3.7: *ANT+ data message format*

3.5.3 Profiles

Each ANT+ device has a profile that contains the network rules relevant to a specific use case (e.g Environment or Heartrate monitor). The profile indicates what type of sensor is used. The sensor specific data that the sensor sends will be found in the payload of the message. Each device can have different data pages, depending on what information the device wants to send.

Example 3.1

For the Environment device profile, three different data pages can be sent from the sensor to another device.

- Data page 1 - Temperature
- Common Page 80 - Manufacturer's Identification
- Common Page 81 - Product Information

The data page 1's table is available in kapitel A. The Common data page 80 transmits the manufacturer's ID, model number, and hardware revision. Common data page 81 transmits the device's software revision and its 32-bit serial number. The common page 80 and 81 is transmitted at a lower rate, ones every 65 pages. The display device can, if needed, send a request to the sensor for a data page.

The payload for the ANT+ has the format shown in Table 3.8:

Byte	Description	Length
0	Data page number	1 Byte
1-7	Sensor Specific Data	7 Bytes

Table 3.8: *ANT+ General Message Format*

3.6 Multi-Channel

Some applications are taking advantage of the multi-channel capabilities of ANT to increase performance. Example of multi-channel applications are:

- **Display/Collector devices:** Devices that receive and display data from a number of sensors. For example, an ANT+ smartphone is receiving several signals, such as heart rate, foot-steeps and GPS signal.
- **Multi-Function Sensors:** Sensors that need to collect data from other sensors, combine the data and send to a master device.
- **Relay Devices:** Some ANT devices are intended to relay information from one location to another, listen to many channels, and then passing the data along to a separate transmit channel.
- **Advanced Network Topologies:** Using multiple channels to enable more complex network designs. Star network or a personal arena network (PANs) can be such networks.

3.6.1 Network example

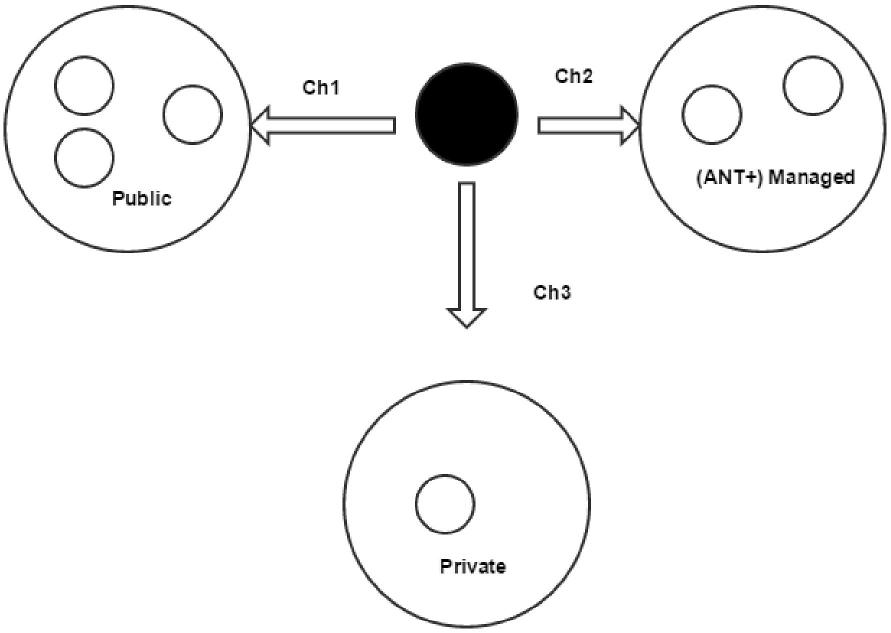


Figure 3.5: *Multi-Network Channels.*

One example of a network that uses Multi-Network Channels is shown in figure 3.5. It could be a cross-country skiing competition. The skiers have a heart rate monitor(in the ANT+ manged network) that's sending data to the black device, the black device send the data to the public network. The only way to communicate between the different networks is via the black device that has access to all the networks.
Adding more channels to a device affects the power consumption. One way to increase power efficiency is by managing channel periods to keep the overall radio usage low. For example, an ANT+ heart rate sensor would send at 1 Hz insted of the standard of 4 Hz. [11]

3.7 Security

Channel encryption can be enabled on an independent channel, not shared on supported devices. When encryption is active, the ANT channel uses 128-bit AES-CTR ,not all chips allows channel encryption. [13]

3.8 ANT File Share

ANT File Share (ANT-FS) is used for transferring files between two ANT devices. The ANT-FS client is typically a mobile device (ex. smartphone, watch or fitness band) that's collecting and storing sensor data. The ANT-FS host is typically a device that wants to download sensor data from an ANT-FS client (ex. PC or smartphone). When a connection initiates between a host and a client, the host sends a link command to the client. Once a link is established, the ANT-FS application specifies what level of authentication is required. When the link has been successfully authenticated, data transport may begin.[12]

In figure 3.6 shows the LINK when it's active after the host sent a Request ANT-FS, and the ANT-FS client sends a "beacon" signal containing the current information about the device. The host opens up the LINK and the transmitting of the FIT-file can proceed:

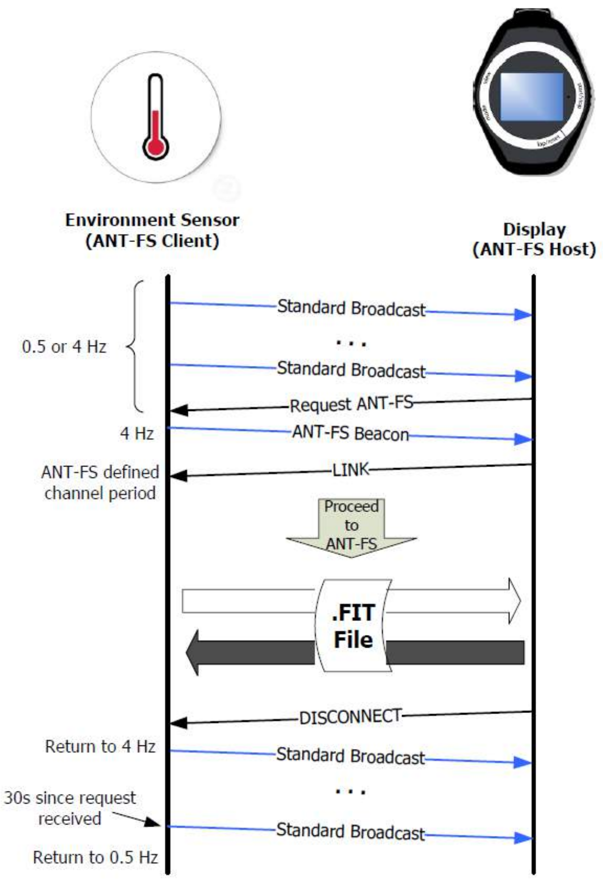


Figure 3.6: *Transmission over ANT-FS.*

3.9 Other technologies

Besides ANT, they are other short-range and low-power protocols. The biggest challenger against ANT is Bluetooth Low Energy (Bluetooth LE or Bluetooth Smart) and ZigBee.

3.10 Bluetooth Low Energy

Bluetooth Low Energy (Bluetooth LE) is a low-power version of Bluetooth wireless technology. Bluetooth LE was design to exploit the existing 2.4Ghz radio band to reduce cost and complexity in existing radio solution. Bluetooth LE transmit power is variable up to 10mW, drawing approximately 15mA at 3 Volt. [21]

Just like ANT+, a lot of new smart devices are on the market. Windows, Apple, Samsung and Motorola they all have support for Bluetooth LE in their new phones and laptops. Garmin, Casio and Nike are some companies that make Smart devices for sport and fitness, supporting Bluetooth LE. [3]

3.10.1 Key Features

For ultra low power consumption, Bluetooth LE has a number of features that are interesting to focus on: [17]

- Interoperability
- Robustness
- Simplicity
- Bit Rates
- Latency
- Range

3.10.2 Interoperability

To ensure that devices whit Bluetooth LE can communicate with other Bluetooth LE devices, the Bluetooth SIG builds strong qualification and interoperability testing processes into the definition of the technology. Bluetooth LE uses the 2,4GHz frequency band, that users can count on is working worldwide. [17]

3.10.3 Robustness

Bluetooth LE uses frequency hopping, to secure a robust transmission between devices. It can be very suitable in environments where multiple devces use the 2,4GHz frequency band. [17]

3.10.4 Simplicity

The Bluetooth LE shares much of the Bluetooth Classic technology. The Bluetooth LE chips can be equal to the size of a coincell, only a few millimeter thicker. [17]

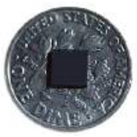


Figure 3.7: A typical size of a Bluetooth LE chip.

3.10.5 Bit Rates

Like ANT the Bluetooth LE is most effcient when transfering a small amount of data (down to 8 bytes). The transfer rate is 1 Mbit/s. [17]

3.10.6 Latency

Bluetooth LE latency for transferring data is generally 6 ms, but can be down to 2,5-3ms. [17]

3.10.7 Range

Bluetooth LE can send data up to ca 50 meters, the Bluetooth Classic range is about 10 meters. [17]

3.10.8 Security

Bluetooth LE use full AES-128 encryption using CCM to provide strong encryption and authentication of data packets. [4]

3.10.9 Packets

Bluetooth LE have one packet structure with two types of packets: advertising and data. The Bluetooth packet begins with Preamble and ends with CRC and has the following format:

Preamble	Access Address	PDU	CRC

Table 3.9: Bluetooth packet format

The packet format:

Name	Length	Description
Preamble	1 Byte	Uses for synchronization
Access Address	4 Bytes	Uses for for physical link identification on every packet for each slave
PDU	2-39 Bytes	Header and Data
CRC	3 Bytes	Uses for the CRC ensures correctness of the data in the PDU on all packets, thus increasing robustness against interference

Table 3.10: *Bluetooth LE data message format*

3.11 ZigBee

ZigBee is based on an IEEE 802.15 standard. ZigBee is a low data rate, low cost and low power consumption wireless protocol. Ziggbee are using 2.GHz (worldwide), 915Mhz (America and Australia) and 868 MHz band (Europe)[5]. ZigBee typical application areas are:[18]

- Home Entertainment and Control
- Wireless sensor networks
- Industrial control
- Medical data collection

3.11.1 Bit Rates

The data rate is 250 kbit/s per channel in the 2.4 GHz band, for the American and Australia 915 MHz band the data rate is 40 kbit/s and for the European 868 MHz band is the data rate 20 kbit/s. [5]

3.11.2 Latency

Ziggbee latency for transferring data is between 20-30ms.[5]

3.11.3 Range

Zigbee can send data up to ca 100 meters, depending on power output and environmental characteristics. A typical device on 1mW are expected to send up to 20 meters. [24]

3.11.4 Security

ZigBee networks are secured by 128 bit symmetric encryption keys. [24]

3.11.5 Packets

The packet structure for Zigbee is the PHY protocol data unit (PPDU).

Preamble	Start-of-packet delimiter	PHYH	PSDU

Table 3.11: *Zigbee packet format(PPDU)*

The packet format consist of:

Name	Length	Description
Preamble	4 bytes	Uses for synchronization
Start-of-packet delimiter	1 byte	Uses for signify end of preamble
PHY Header	1 byte	Specify length of PSDU
PSDU	Up to 127 bytes	PHY layer payload

Table 3.12: *Zigbee data message format*

3.12 Comparison ANT+, Bluetooth LE and Zigbee

Making a comparison between two different technologies could in itself be a completely separate report. Here are the things that is relevant to the ANT+, Bluetooth LE ant Zigbee, various functionalities that may be crucial for the end-user of the technology that best suits their purpose.

Description	ANT+	Bluetooth	Zigbee
Data Rate [20]	1Mbps	1Mbps	0,25Mbps
Latency [20]	Zero	2.5ms	20-30ms
Maximum Peak Current [20]	17mA	12-15mA	30-40mA
Range [8]	10m	50m	100m

Table 3.13: *Comparisons Of ANT+, BT LE and Zigbee Protocols*

Both ANT and Bluetooth LE promise Ultra low peak, average and idle mode power consumption. The both have the ability to run on a single coin cell for years, they are immunity to interference, and are supports by multi vendors. The usage between ANT and Bluetooth separates them however. Table 3.14 shows use case comparison.

Wireless usage scenario		**ANT+**	**BT LE**
1 sensor	$\rightarrow$ 1 user display	$\surd$	$\surd$
7 sensors	$\rightarrow$ 1 user display	$\surd$	$\surd$
50 sensors	$\rightarrow$ 1 user display	$\surd$	χ
1 sensor	$\rightarrow$ 2+ user display	$\surd$	χ
1 user display	$\rightarrow$ 1 user display	$\surd$	χ
1 sensor	$\rightarrow$ 1 sensor	$\surd$	χ
1 sensor	$\rightarrow$ 2+ sensors	$\surd$	χ

Table 3.14: *Use case comparison*

4 Database

One of the important features of the assignment of this master project is to store data in a database. To save the data in a database should be flexible, secure and reliable. For the case of saving the data from the application to use in other applications or upload to the internet the Flexible and Interoperable Data Transfers (FIT) protocol is used, FIT is a format that a lot of devices with ANT+ support and manage to make use of. In the design the data is also stored in a simple SQLite database, for the simplicity to use a standardize database.

4.1 FIT

Flexible and Interoperable Data Transfers (FIT) protocol is a format designed for storing data from sport, fitness and health devices. An example of the FIT-file is shown in the appendix. The following example illustrates how the FIT protocol is used to transfer training data between devices and to share the training data to the internet: [10]

1. ANT+ Sensors measure parameters such as heart rate and running speed
2. Data is broadcast in real time, using interoperable ANT+ data formats
3. Session events and real time activity data is collected and saved into a FIT file on a display device
4. The FIT file is transferred to the PC using ANT File Share (ANT-FS)

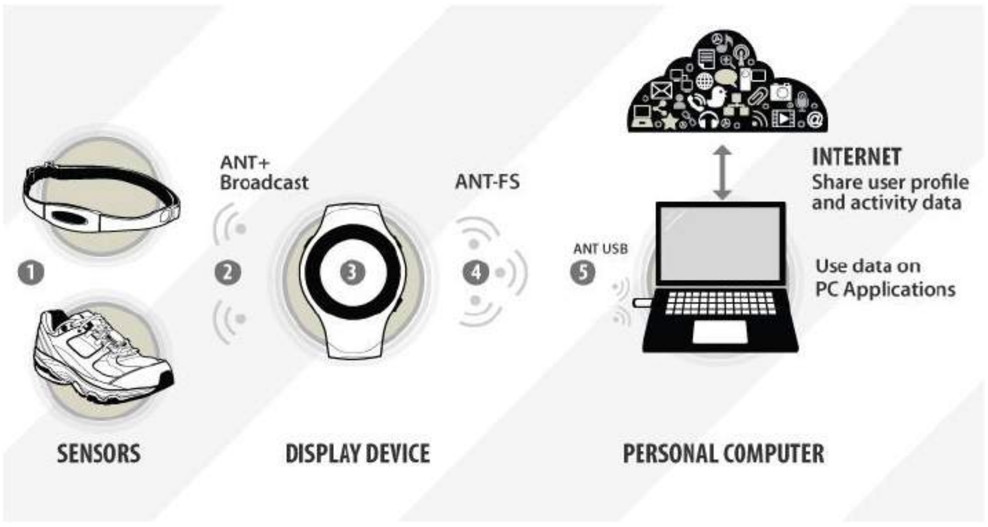


Figure 4.1: *Data flow between devices using FIT protocol [13]*

4.2 FIT File Protocol

The FIT file protocol defines the process for which profiles are implemented and transferred. Figure 4.2 shows an overview of a typical ANT+ device. The FIT Encoder encodes both Device specific FIT messages (can be custom messages defined by the manufacture) and incoming data like settings, events, sensor data, etc. The file is transferred to another device where it is decoded. [10]

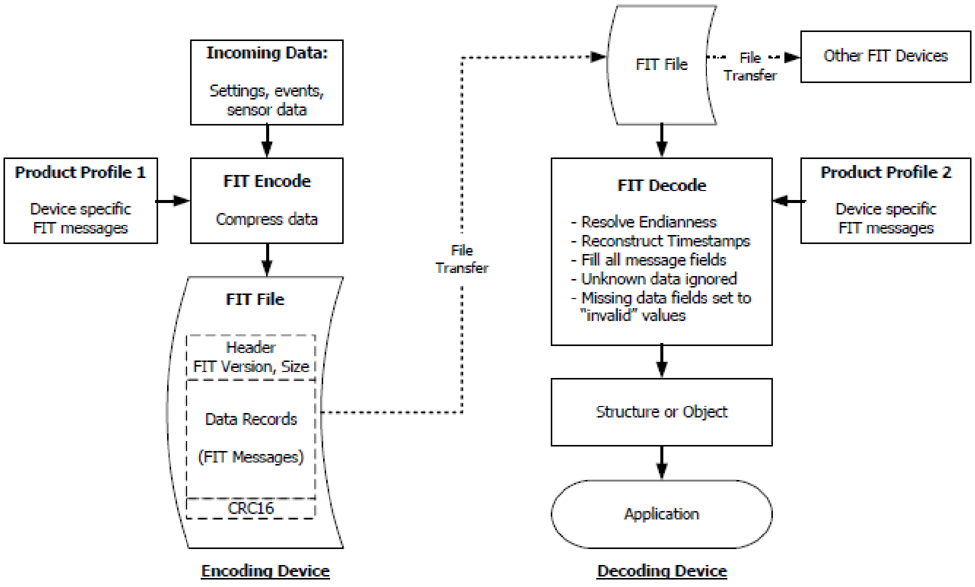


Figure 4.2: *Overview of the FIT File protocol*

4.3 FIT File Structure

The FIT file has a structure that consists of a File Header, Data Records and a 2-byte CRC. [10]

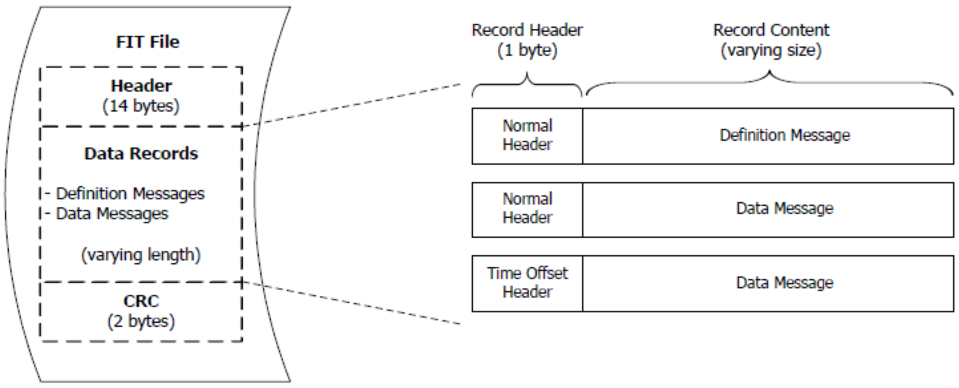


Figure 4.3: *The FIT file with data record types*

To properly encode/decode a FIT-file the following must be included: Header, Data Records and 2 bytes CRC. [10]

4.3.1 Header

The header file provides information of the FIT file. Below follows a description of the header file:

Byte	Parameter	Description	Size
0	Header Size	Indicates the length of this file header including header size.	1 Byte
1	Protocol Version	Protocol version number as provided in SDK	1 byte
2	Profile Version LSB	Profile version number as provided in SDK	1 byte
3	Profile Version MSB		
4	Data Size LSB	Length of the Data Records section in bytes. Does not inclued Header or CRC	4 Bytes
5	Data Size		
6	Data Size		
7	Data Size MSB		
8	Data Type Byte[0]	ASCII values for ".FIT". A FIT binary file opened with a text editor will contain a readable ".FIT" in the first line.	4 bytes
9	Data Type Byte[1]		
10	Data Type Byte[2]		
11	Data Type Byte[3]		
12	CRC LSB	This field is optional.	2 Bytes
13	CRC MSB		

Table 4.1: *Description of File Header*

4.3.2 Data Records

In a simplified way the Data Records contain two different types of messages:

- Definition message: describes the architecture, format, and fields of upcoming data messages
- Data Message: contains data that is formatted according to a preceding definition message

As an example: a sensor that receives heart beat rate, the first message in the FIT file data records is the Definition message that inform about the type of data. Appendix A shows an example of how data is stored in the FIT files. [10]

4.3.3 CRC

The CRC part of the FIT file is stored into the last 2 bytes of the file. [10]

4.4 SQLite

SQLite is an embedded SQL database engine. SQLite was released in 2000 by D. Richard Hipp when he was working for General Dynamics on a program for the US Navy. The goals of the design was to allow the program to operate without installing a database management system. [19] SQLite is free to use, both commercially or privateley. Big company's like Apple, Google and Microsoft have implemented and use SQLite in different projects[23]. Because SQLite is embedded into every Android device, it is a flexible and powerful tool to use and easy to implement. Different from other SQL databases, SQLite does not have a separate server process. The SQlite database file reads and writes directly to ordinary disk files, it can not be run in the background as a process but executes only when asked to execute [19]. According to SQLite organization, they believe that there are more copies of SQLite than any other SQL database. [22].

5 Implementation of the Android application

This chapter will describe the implementation of the system. We will explain the data flow and functions between the components, system design, graphical interface, the ANT+ communication and the design solutions.

5.1 Development environment

The decision for development environment was to use Android Studio for the Android application. Android Studio has a lot of plugins that are used for the implementation, plugins that are useful for git, unit tests and the Android Virtual Device for running of the application.

5.1.1 Android SDK

Android Software Development Kit is the required SDK to develop applications for Android. It provides API libraries, tutorials, an emulator and a debugger for developers.

5.1.2 Android Studio

Android Studio is an integrated development environment (IDE) for developing applications for the Android platform. It's based on JetBrains' IntelliJ IDEA software.

5.1.3 GIT

Git is a distributed revision control system. Git has been used for version control and for backup of the code base.

5.1.4 FitCSVTool

FitCSVTool is a tool for converting the .fit files to .csv files. It is useful for the type of information and data that the .fit file contains.

5.1.5 Android Virtual Device

To simulate android on a computer an emulator is used. Android Virtual Device (AVD) is an emulator to simulate the Android OS, and the application on it.

5.1.6 ANTAndroidBridge

In order to use the AVD with the ANT-USB the ANT Android Bridge is required. The ANT Android Bridge is a bridge between the AVD and Windows. On the AVD that runs android you only need an application that speaks with the ANT Android Bridge in order to work. When connected the AVD can use the ANT-USB as it was integrated in the hardware as a simulation of a phone/tablet or any other system that runs Android.

5.1.7 Hardware

A Smartphone that supports ANT+, is used in this case, a Samsung Galaxy S4. For the ANT+ to work the applications ANT Radio Service, ANT Usb Service and ANT+ Plugins need to be installed (often pre installed by the manufacture).

5.2 System design

The system design is based on the requirements of the application. The requirements are obtained by the product backlog based on the user stories.

- A user should be able to choose which sensors to connect to.
- A user should be able to receive and display data from two sensors.
- A user should be able to save data into a database.
- A user should be able to send files over ANT-FS.

5.2.1 System overview

The main architecture of the system is based on multiple layers to separate the different functions of the application. Each layer has its own functionality and controls different functions of the application. The three layers used are: The presentation layer, the function layer and a data layer.

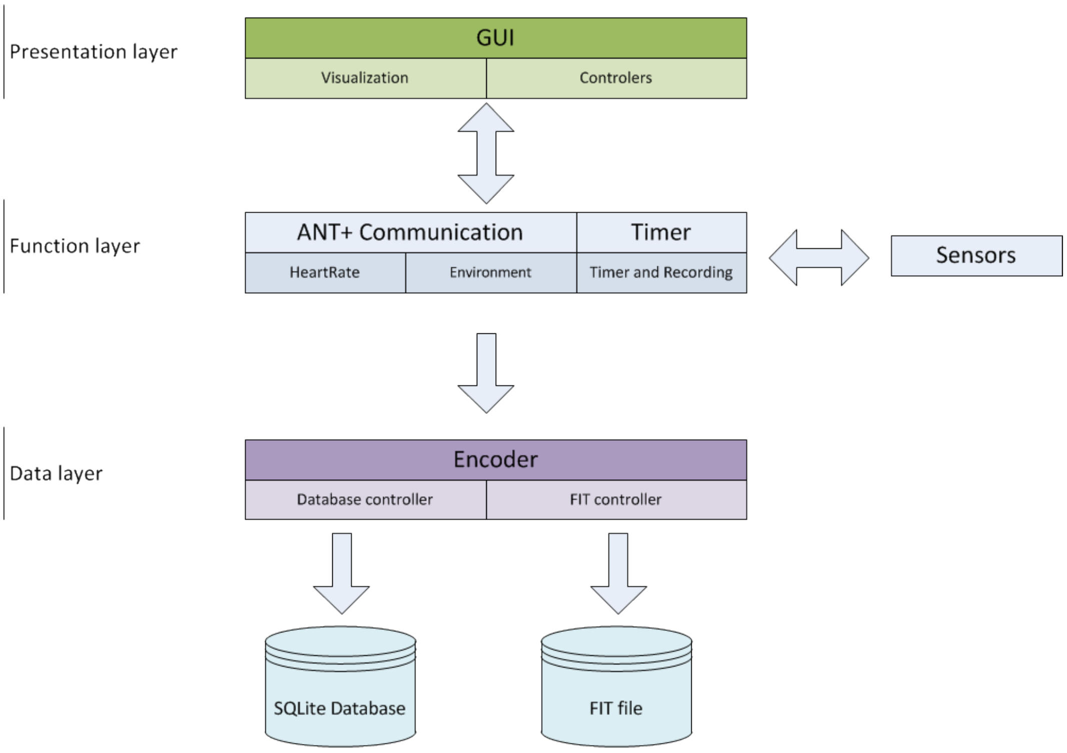


Figure 5.1: *System overview*

5.3 Presentation layer

The presentation layer controls the visualization of the ANT+ sensor data. It shows the temperature and the current heart rate when the sensors are connected. The presentation layer also has buttons to interact with the application. All the interactions are taken care of by the presentation layer. The graphical user interface is shown in figure 5.2.

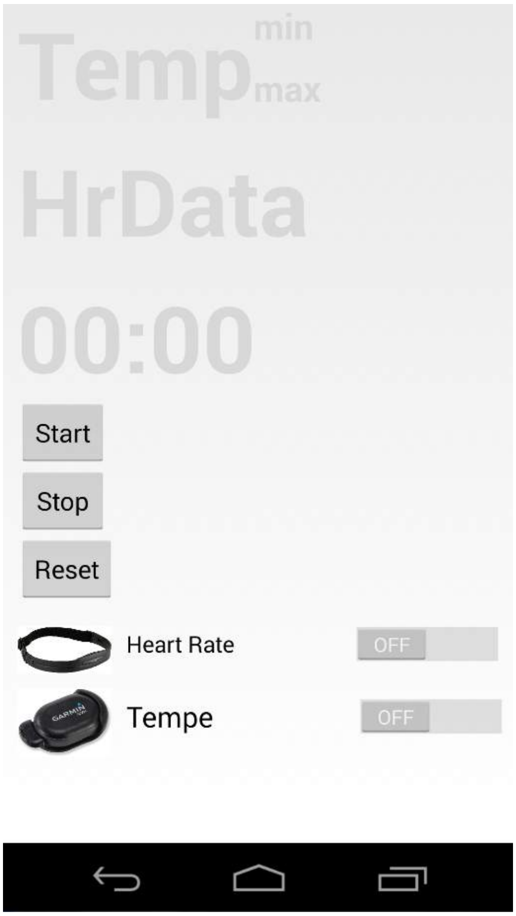


Figure 5.2: *Homescreen application window.*

5.3.1 Class Diagram

The figure 5.3 shows the class diagram for the presentation layer. The presentation layer consists of the the Main class.

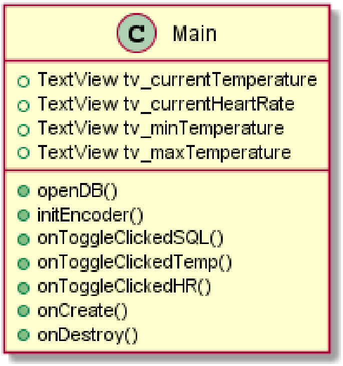

C Main
TextView tv_currentTemperature
TextView tv_currentHeartRate
TextView tv_minTemperature
TextView tv_maxTemperature
openDB()
initEncoder()
onToggleClickedSQL()
onToggleClickedTemp()
onToggleClickedHR()
onCreate()
onDestroy()

Figure 5.3: *Homescreen application class.*

5.3.2 GUI

The GUI (Graphical user interface) is designed to have all functionality in only one frame, for easy overview. The home screen of the application is divided in two parts, the data visualization of the sensors and control-part of the application.

5.3.3 Visualization

The upper part of the screen is the visualization part, shows in figure 5.4:

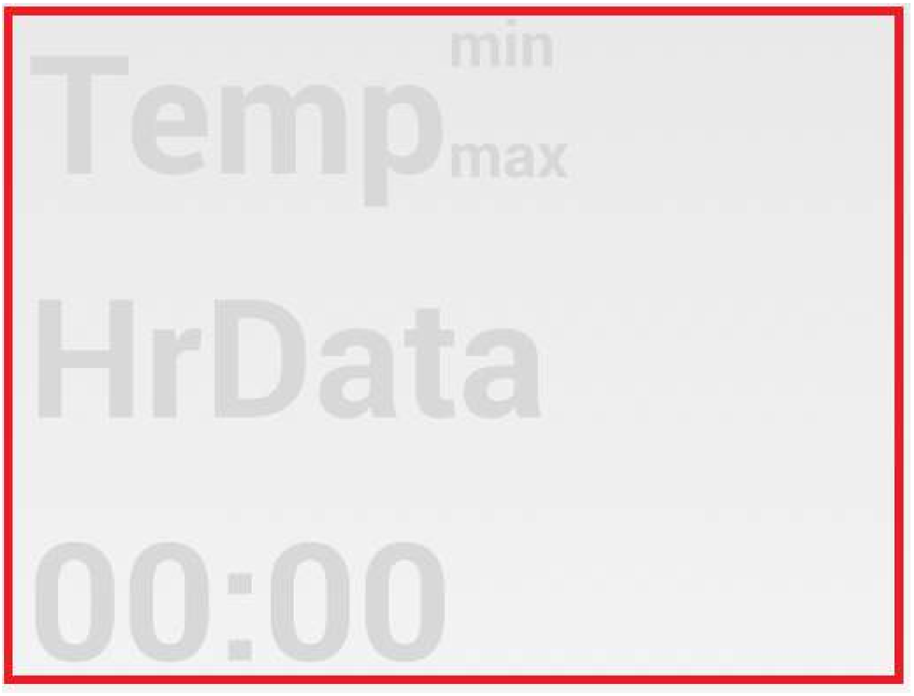


Figure 5.4: *Visualization part of application.*

The Temp-field is showing the current temperature in Celsius from the sensor when the temperature sensor is connected. Min and max are showing minimum and maximum temperature for the latest 24 hours. The HrData-field is showing the current heart rate data when the heart rate sensor is connected. The Timer shows seconds and minutes.

5.3.4 Controllers

The Controller part of the application home screen is used to interact with the user. It also controls what devices that can be connected and is the controllers for the timer, shown in figure 5.6.

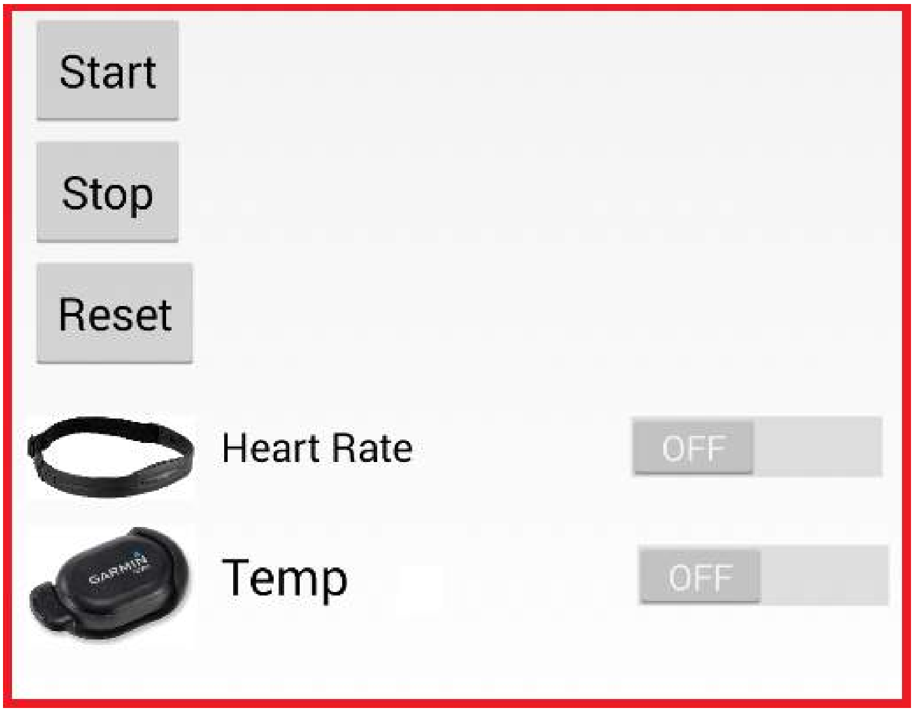


Figure 5.5: *Controller part of the application.*

5.3.5 Timer controls

The timer has two functions, it show current time and saves data. To start the timer the user presses the start button. Two things happen, a timer starts and a FIT-file is created. Every second the information of temp and heartrate data is written to the FIT-File. On stop, the timer stops and the FIT-file closes and can be used in different applications. The "Reset" resets the timer to zero.

5.3.6 Heart rate

The heartrate switch starts the searching of a heart rate sensor. When connected the heart rate data is shown in the HrData-field.

5.3.7 Temp

The temp switch starts the searching of a temperature sensor. When connected the heart rate data is shown in the Temp-field.

5.4 Function layer

The ANT+ communication between the application and the sensors is controlled by the Function layer. The system is designed to make it easy to add more ANT+ sensors and functionality. Each ANT+ module runs independently. The function layer sends data both to the data layer and the presentation layer for visualization of both sensor data and the timer.

The ANT+ sensors are connected to the android application with help from the Android ANT+ API. The ANT+ sensor and the application use device paring to establish a connection between each other. The Timer controls both the time-keeping and the recording of the data to the data layer. When the timer is ticking the encoder in data layer saves the data to each database.

Below is a brief description how the application connects to a sensor and gets sensor data:

1. User toggles a switch.
2. A request is sent to access the sensor.
3. The application is searching for a sensor.
4. The sensor connects with the application
5. The application is in tracking state and can now get data from the sensor
6. The application is now subscribing on any events from the sensor
7. The data from the sensor is set when the application receives data.

Figure 5.6 presents the sequence diagram of what happens when the user is connecting to a sensor and gets the data from the sensor.

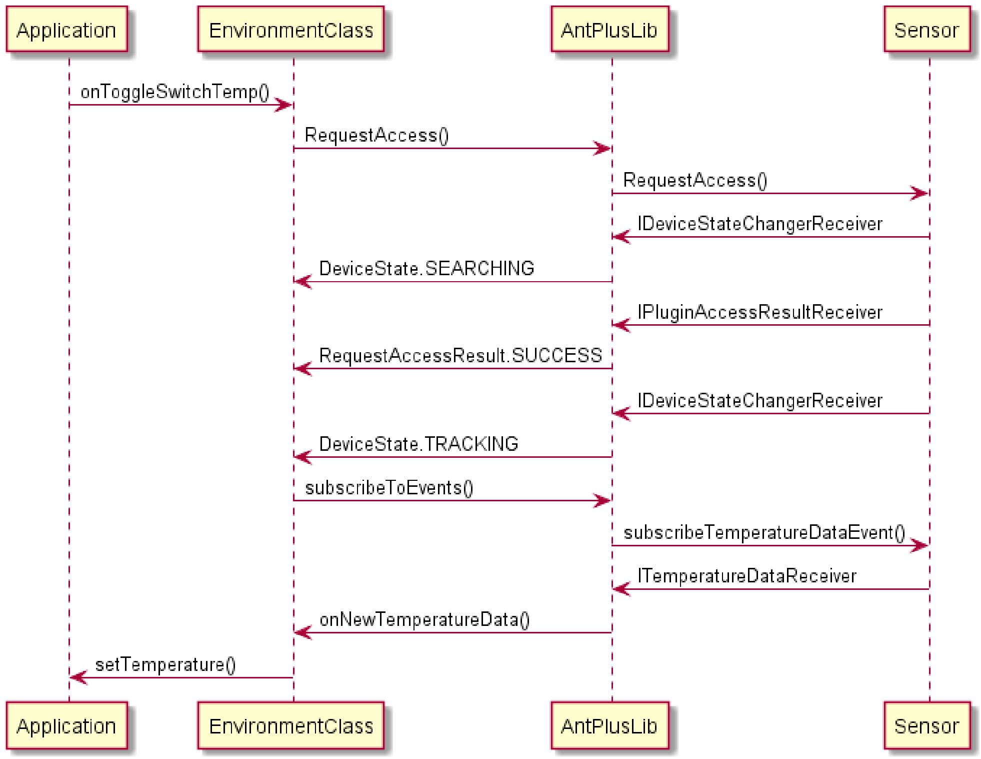


Figure 5.6: *Sequence diagram.*

When the user toggles the switch for the environment sensor the application requests to access a sensor. The device state is set to SEARCHING, searching for any sensors that can be connected. When a sensor is connected the Request Access Result is set to SUCCESS and the device state to TRACKING. The application is connected to the sensor and ready to receive data. The application now listens for events from the sensor, and receives data from a temperature data event. Each time a new temperature data is received the application updates and prints the recent data out on the screen.

5.4.1 Class diagram

The function layer consists of the Environment, HeartRate, Timer classes and the ANT+ Android library package that is needed for the ANT+ part of the application to work. The Environment and HeartRate classes use the antplus.pccbase package to connect and disconnect to the sensors. The antplus.pcc package is used to get data from the sensors when connected. The class diagram for the function layer is shown in figure 5.7.

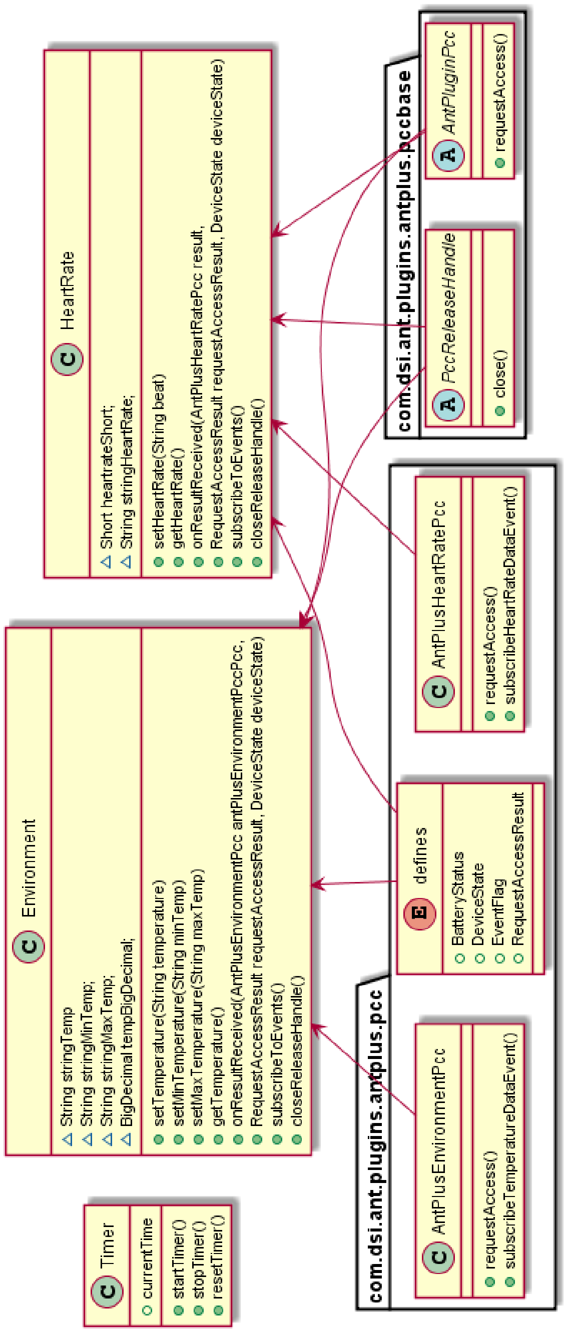

HeartRate
Short heartrateShort;
String stringHeartRate;
setHeartRate(String beat)
getHeartRate()
onResultReceived(AntPlusHeartRatePcc result, RequestAccessResult requestAccessResult, DeviceState deviceState)
subscribeToEvents()
closeReleaseHandle()
Environment
String stringTemp
String stringMinTemp;
String stringMaxTemp;
BigDecimal tempBigDecimal;
setTemperature(String temperature)
setMinTemperature(String minTemp)
setMaxTemperature(String maxTemp)
getTemperature()
onResultReceived(AntPlusEnvironmentPcc antPlusEnvironmentPccPcc, RequestAccessResult requestAccessResult, DeviceState deviceState)
subscribeToEvents()
closeReleaseHandle()
Timer
currentTime
startTimer()
stopTimer()
resetTimer()
com.dsi.ant.plugins.antplus.pccbase
AntPluginPcc
requestAccess()
PccReleaseHandle
close()
com.dsi.ant.plugins.antplus.pcc
AntPlusHeartRatePcc
requestAccess()
subscribeHeartRateDataEvent()
defines
BatteryStatus
DeviceState
EventFlag
RequestAccessResult
AntPlusEnvironmentPcc
requestAccess()
subscribeTemperatureDataEvent()

Figure 5.7: *Function layer class diagram.*

5.5 Data Layer

The data layer is storing the data in the database or in the FIT files. The Encoder is saving the data to the databases. One controller is used for the SQLite database, and another controller to save data in FIT-files.

5.5.1 Class diagram

The data layer consists of the DatabaseController, FitController, TimeController and DatabaseHelper classes. The DatabaseController and DatabaseHelper classes imports the android.database.sqlite package. The FitController and TimeController imports the com.garmin.fit package. The com.garmin.fit package is used to encode and decode FIT-files. The class diagram for the Encoder is shown in figure 5.8 and figure 5.9.

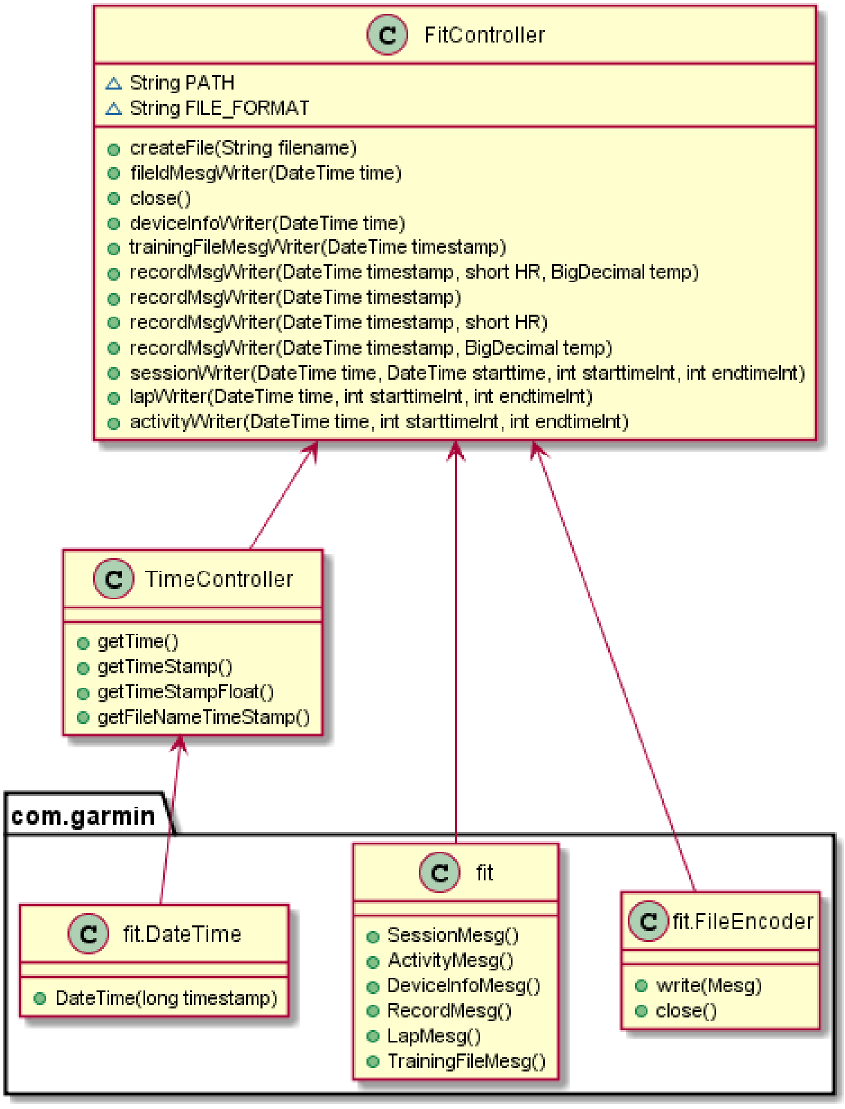

FitController
String PATH
String FILE_FORMAT
createFile(String filename)
fileIdMesgWriter(DateTime time)
close()
deviceInfoWriter(DateTime time)
trainingFileMesgWriter(DateTime timestamp)
recordMsgWriter(DateTime timestamp, short HR, BigDecimal temp)
recordMsgWriter(DateTime timestamp)
recordMsgWriter(DateTime timestamp, short HR)
recordMsgWriter(DateTime timestamp, BigDecimal temp)
sessionWriter(DateTime time, DateTime starttime, int starttimeInt, int endtimeInt)
lapWriter(DateTime time, int starttimeInt, int endtimeInt)
activityWriter(DateTime time, int starttimeInt, int endtimeInt)
TimeController
getTime()
getTimeStamp()
getTimeStampFloat()
getFileNameTimeStamp()
com.garmin
fit.DateTime
DateTime(long timestamp)
fit
SessionMesg()
ActivityMesg()
DeviceInfoMesg()
RecordMesg()
LapMesg()
TrainingFileMesg()
fit.FileEncoder
write(Mesg)
close()

Figure 5.8: *Function layer class diagram - FitController.*

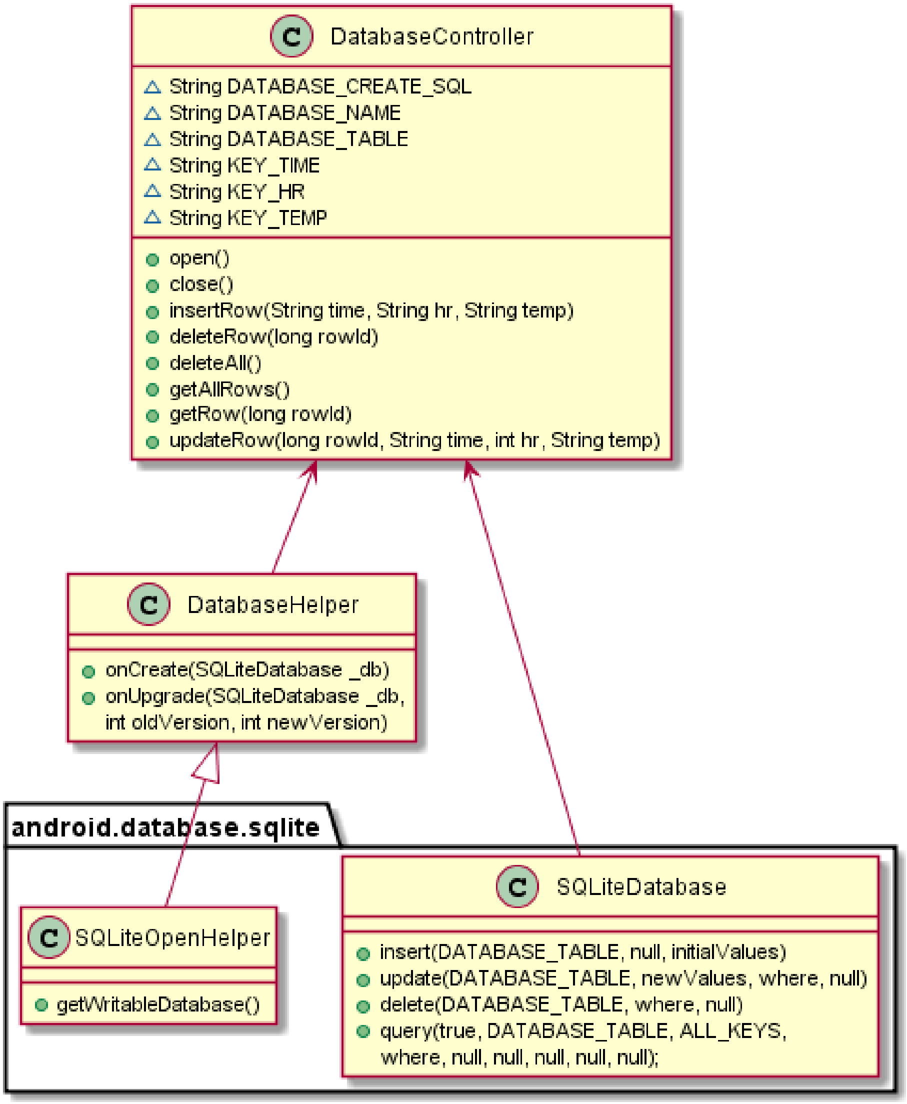

DatabaseController
△ String DATABASE_CREATE_SQL
△ String DATABASE_NAME
△ String DATABASE_TABLE
△ String KEY_TIME
△ String KEY_HR
△ String KEY_TEMP
● open()
● close()
● insertRow(String time, String hr, String temp)
● deleteRow(long rowId)
● deleteAll()
● getAllRows()
● getRow(long rowId)
● updateRow(long rowId, String time, int hr, String temp)
DatabaseHelper
● onCreate(SQLiteDatabase _db)
● onUpgrade(SQLiteDatabase _db, int oldVersion, int newVersion)
android.database.sqlite
SQLiteOpenHelper
● getWritableDatabase()
SQLiteDatabase
● insert(DATABASE_TABLE, null, initialValues)
● update(DATABASE_TABLE, newValues, where, null)
● delete(DATABASE_TABLE, where, null)
● query(true, DATABASE_TABLE, ALL_KEYS, where, null, null, null, null, null);

Figure 5.9: *Function layer class diagram - DatabaseController.*

5.5.2 FIT

The FIT-File is divided into three parts, Creating of the file, recording data and the closing of the file.

5.5.2.1 Create

At the creation of the file the FIT-file writes a "File ID Message". The "File ID Message" can hold information about the type of product, manufacturer, type of file, time created and much more. In this thises the "File ID Message" contains as little information as possible, just created date and type.

5.5.2.2 Recording

At the recording part the FIT-files store the record message with timestamp, heart rate value and the temperature.

5.5.2.3 Closing

The closing of the file, the FIT-file stores a Device info message with a timestamp, a lap message with the total timer time, a Session message with start time and total time, and finally an activity message with the total time.

6 Implementation of the Windows application

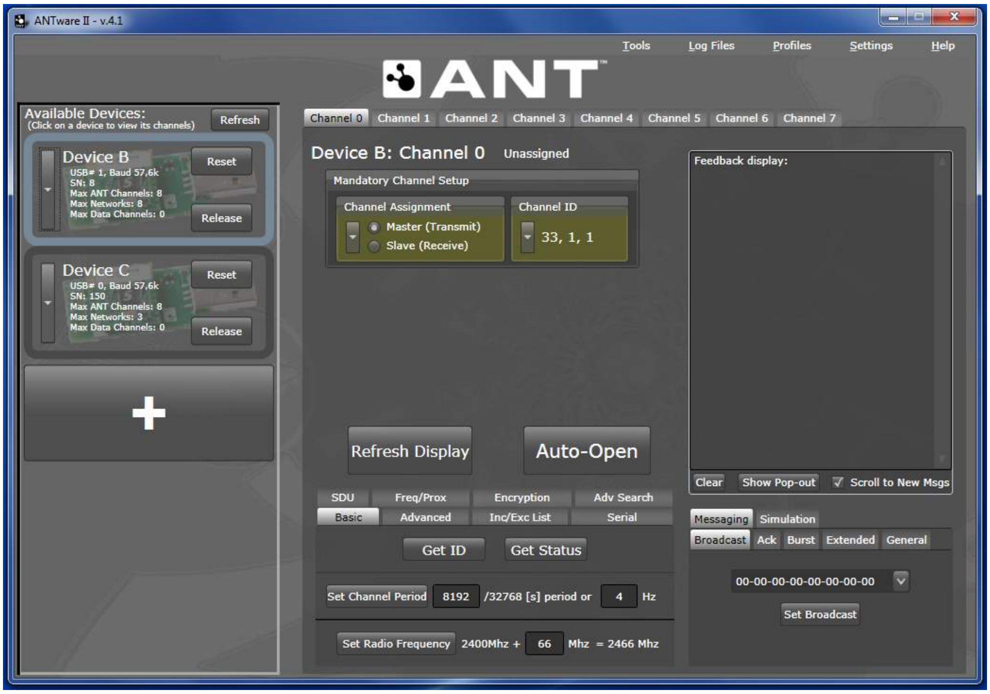
This chapter will describe the implementation of the Windows application. This chapter explains the data flow and functions between the components, system design, graphical interface, the ANT+ communication and the design solutions.

6.1 Development environment

The decision for development environment landed on Visual Studio Express 2013 for the Windows application. Visual studio is used to develop applications for Windows. Visual studio has a design tool implemented to make it easy to develop forms/windows for the applications that the developer wants to build.

6.1.1 ANTware II

ANTware II is an application used for controlling ANT devices. It is a good tool for learning and explore ANT. It is easy to set up a ANT network and understanding how ANT works. It can be used with an ANT USB stick.



Device B
USB# 1, Baud 57,6k
SN: 8
Max ANT Channels: 8
Max Networks: 8
Max Data Channels: 0

Device C
USB# 0, Baud 57,6k
SN: 150
Max ANT Channels: 8
Max Networks: 3
Max Data Channels: 0

Device B: Channel 0 Unassigned

Channel Assignment: [✓ Master (Transmit)] Slave (Receive)

Channel ID: 33, 1, 1

Set Channel Period 8192 /32768 [s] period or 4 Hz

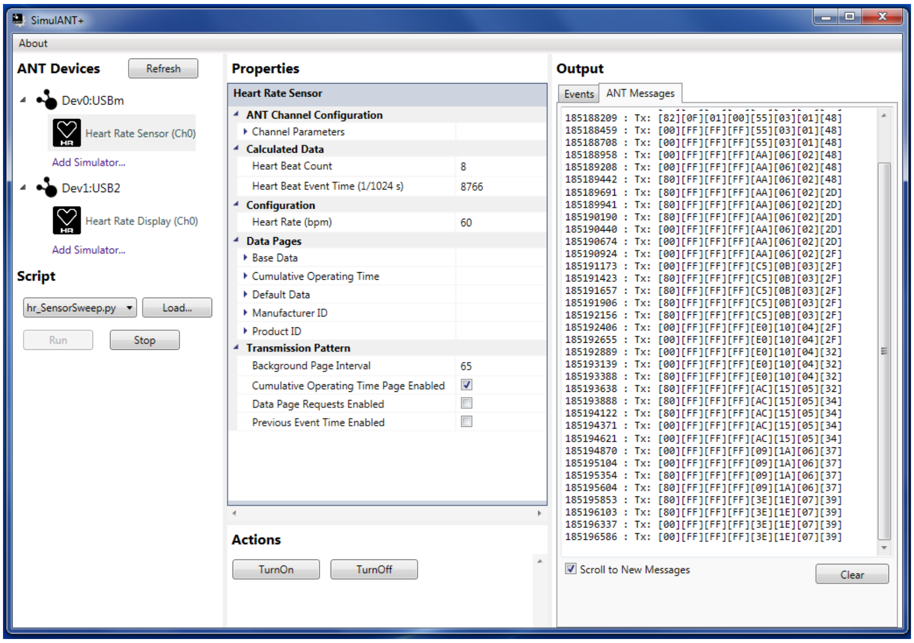
Set Radio Frequency 2400Mhz + 66 Mhz = 2466 Mhz

✓ Scroll to New Msgs

00-00-00-00-00-00-00-00

Figure 6.1: *ANTware II application Window.*

6.1.2 SimulANT+



Heart Rate Sensor	
ANT Channel Configuration	
Channel Parameters	
Calculated Data	
Heart Beat Count	8
Heart Beat Event Time (1/1024 s)	8766
Configuration	
Heart Rate (bpm)	60
Data Pages	
Base Data	
Cumulative Operating Time	
Default Data	
Manufacturer ID	
Product ID	
Transmission Pattern	
Background Page Interval	65
Cumulative Operating Time Page Enabled	✓
Data Page Requests Enabled	☐
Previous Event Time Enabled	☐

```
185188209 : Tx: [82][0F][01][00][55][03][01][48]
185188459 : Tx: [00][FF][FF][FF][55][03][01][48]
185188708 : Tx: [00][FF][FF][FF][55][03][01][48]
185188958 : Tx: [00][FF][FF][FF][AA][06][02][48]
185189208 : Tx: [00][FF][FF][FF][AA][06][02][48]
185189442 : Tx: [80][FF][FF][FF][AA][06][02][48]
185189691 : Tx: [80][FF][FF][FF][AA][06][02][2D]
185189941 : Tx: [80][FF][FF][FF][AA][06][02][2D]
185190190 : Tx: [80][FF][FF][FF][AA][06][02][2D]
185190440 : Tx: [00][FF][FF][FF][AA][06][02][2D]
185190674 : Tx: [00][FF][FF][FF][AA][06][02][2D]
185190924 : Tx: [00][FF][FF][FF][AA][06][02][2F]
185191173 : Tx: [00][FF][FF][FF][C5][0B][03][2F]
185191423 : Tx: [80][FF][FF][FF][C5][0B][03][2F]
185191657 : Tx: [80][FF][FF][FF][C5][0B][03][2F]
185191906 : Tx: [80][FF][FF][FF][C5][0B][03][2F]
185192156 : Tx: [80][FF][FF][FF][C5][0B][03][2F]
185192406 : Tx: [00][FF][FF][FF][E0][10][04][2F]
185192655 : Tx: [00][FF][FF][FF][E0][10][04][2F]
185192889 : Tx: [00][FF][FF][FF][E0][10][04][32]
185193139 : Tx: [00][FF][FF][FF][E0][10][04][32]
185193388 : Tx: [80][FF][FF][FF][E0][10][04][32]
185193638 : Tx: [80][FF][FF][FF][AC][15][05][32]
185193888 : Tx: [80][FF][FF][FF][AC][15][05][34]
185194122 : Tx: [80][FF][FF][FF][AC][15][05][34]
185194371 : Tx: [00][FF][FF][FF][AC][15][05][34]
185194621 : Tx: [00][FF][FF][FF][AC][15][05][34]
185194870 : Tx: [00][FF][FF][FF][09][1A][06][37]
185195104 : Tx: [00][FF][FF][FF][09][1A][06][37]
185195354 : Tx: [80][FF][FF][FF][09][1A][06][37]
185195604 : Tx: [80][FF][FF][FF][09][1A][06][37]
185195853 : Tx: [80][FF][FF][FF][3E][1E][07][39]
185196103 : Tx: [80][FF][FF][FF][3E][1E][07][39]
185196337 : Tx: [00][FF][FF][FF][3E][1E][07][39]
185196586 : Tx: [00][FF][FF][FF][3E][1E][07][39]
```

[✓] Scroll to New Messages

Figure 6.2: *SimulANT+ application window.*

SimulANT+ is an application used for simulating ANT+ devices and displays. It is a good tool to simulate a sensor or a display. The application have a lot of different device profiles that can be useful to simulate a sensor if needed.

6.1.3 Hardware

A personal is used computer that runs Windows 7 with a ANT-USB.

6.2 System design

The system design is based by the requirements of the application. The same requirements as for the android application. The requirements are obtained by the product backlog based on the user stories, see chapter 5.2.1.

6.2.1 System overview

The main architecture of the system is based on two layers to separate the different functions of the application. Each layer has its own functionality and controls different functions of the application. The two layers used are: The Configuration layer and a data layer.

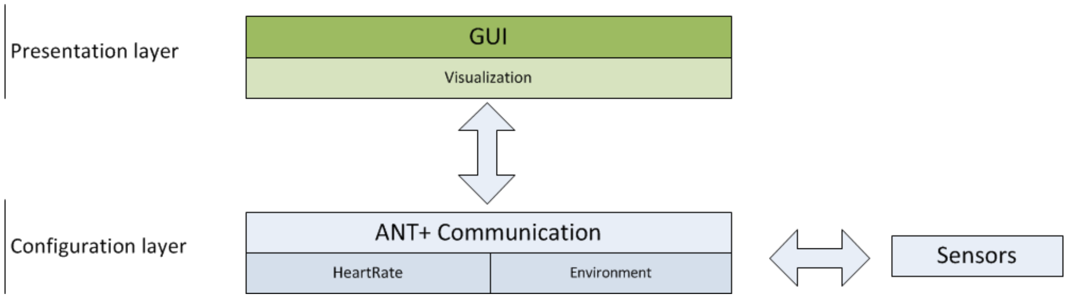


Figure 6.3: *System overview*

6.3 Application

The application is divided into two parts. The Configuration part and the Data part. Each tab controls each part. The graphical user interface is shown in figure 6.4.

6.3.1 Configuration

In the configuration the user can select the usb port of the ANT-USB stick. When selected the user connects with the USB. After connecting the user can open the ANT+ channel. The difference between the Windows application and the Android application is that the Windows application does not have a device paring of sensors. The application just listens on all traffic to the ANT+ channel (2457 Mhz). If the sensor is a environment sensor or an heat rate sensor the data will be shown in the application.

Figure 6.5 shows the configuration part of the application:

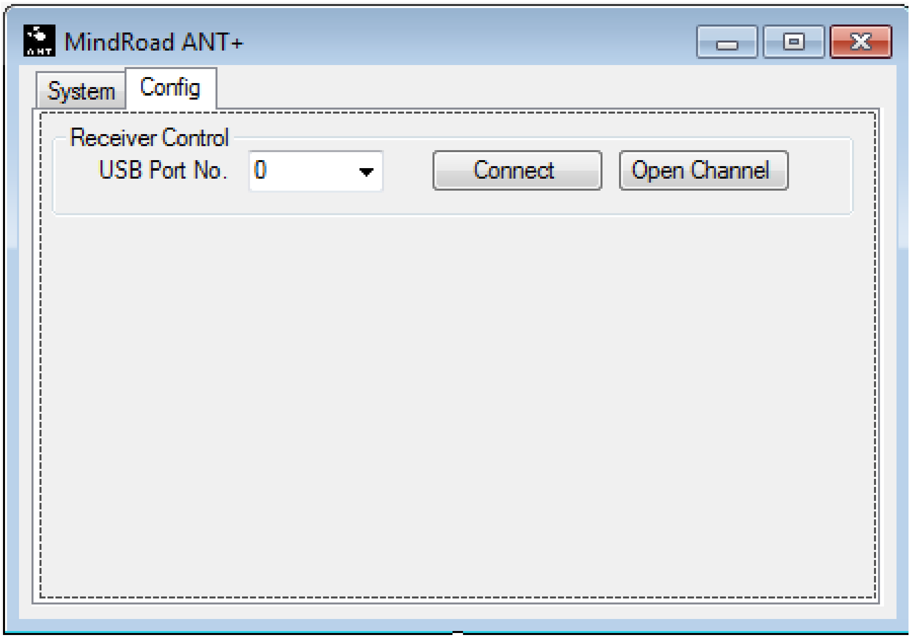


Figure 6.4: *Configuration part of application.*

In the Configuration tab the user can choose which USB port to use, then connect to the USB port and open the ANT+ channel. The reason for choosing this solution to always listen to all the data, is that it was easy to implement in the application, and for testing the two different options for how to read data from ANT+. In this solution, it is the application itself that chooses which data it wants to show. One disadvantage is that if there are multiple sensors that send the same data of the same type, such as Heart Rate Sensors the application does not take this into account, as long as there is a heart rate sensor, then the data will be presented independently which sensor it is.

6.3.2 ANT+ Data

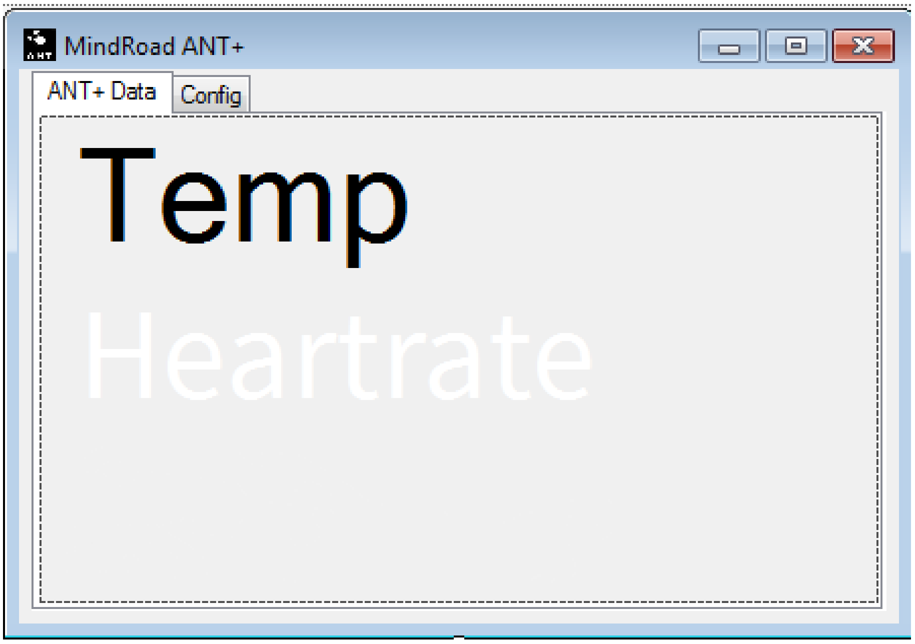


Figure 6.5: *Homescreen windows application.*

The ANT+ Data tab is showing when the ANT+ USB stick is connected to the Temp and Heartrate. The Temp-field is showing the current temperature in Celsius from the sensor when the Tempe sensor is connected. The Heartrate-field is showing the heart rate data when the heart rate sensor is connected.

7

Results

In this section the results and the evaluation will be presented.

7.1 Requirements

The requirements for the system described in section 5 the was following:

- A user should be able to choose which sensors to connect to.
- A user should be able to receive and display data from two sensors.
- A user should be able to save data into a database.
- A user should be able to send files over ANT-FS.

For the Android application all of the requirements except requirement; A user should be able to send files over ANT-FS, was implemented. For the windows application; Save data into a database and sending files over ANT-FS was not implemented. The main focus of development was on the Android application, therefore it has more functionality.

7.2 System design

To use the multiple layers as a model for the system design, is is easy to add or remove functionality. By using this pattern, the visual representation is separated from the data and function layers. By having a well-structured architecture model implemented, the system facilitates further extensions or changes.

7.3 Connecting a sensor

Connecting a sensor with the application is made through pulling the slider from OFF to ON for the sensor to connect. Figure 7.1 shows an example when searching for sensors that can be connect with the application. Since each sensor is written as a separate module, it is easy to extend the application and add other modules to the application. Figure 7.2 shows the application up and running. Both sensors are connected and display the data of the current value.

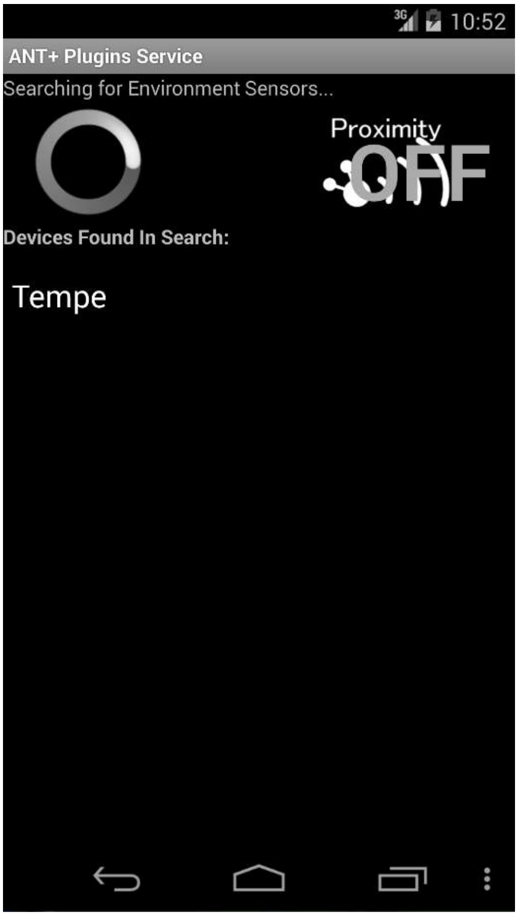


Figure 7.1: *Searching for sensors.*

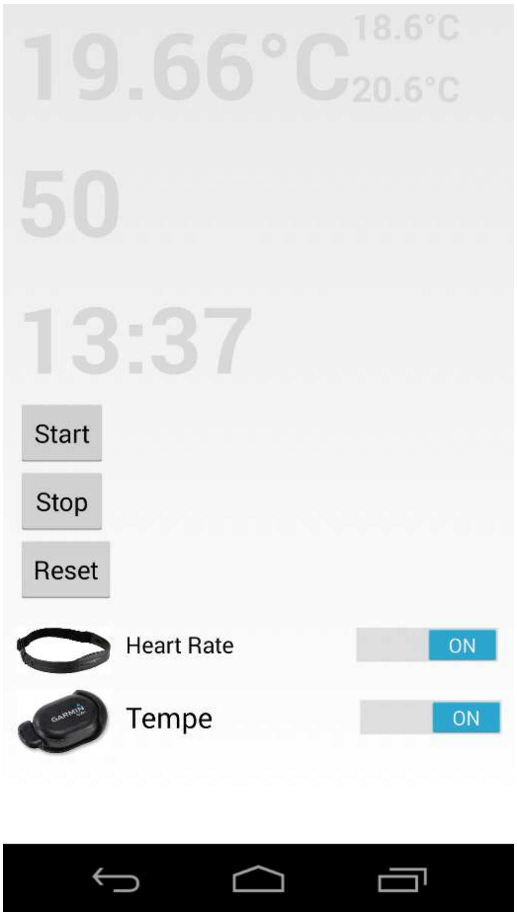
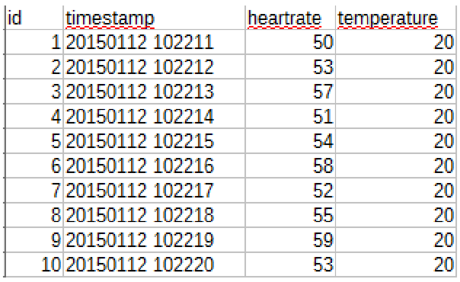


Figure 7.2: Application running.

7.4 Database

7.4.1 SQLite

The SQLite table is really simple. It contains the columns id, timestamp, heartrate and temperature. The id is an unique auto increment integer for each row in the table. timestamp is a String with date and time. heartrate and temperature is just the value of the data. The table is shown in figure 7.3.

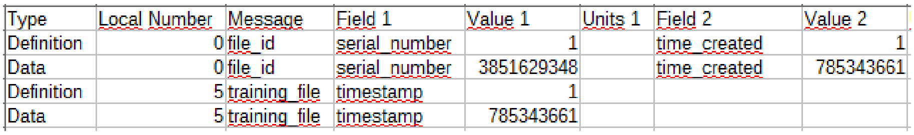


id	timestamp	heartrate	temperature
1	20150112 102211	50	20
2	20150112 102212	53	20
3	20150112 102213	57	20
4	20150112 102214	51	20
5	20150112 102215	54	20
6	20150112 102216	58	20
7	20150112 102217	52	20
8	20150112 102218	55	20
9	20150112 102219	59	20
10	20150112 102220	53	20

Figure 7.3: *SQLite data.*

7.4.2 FIT

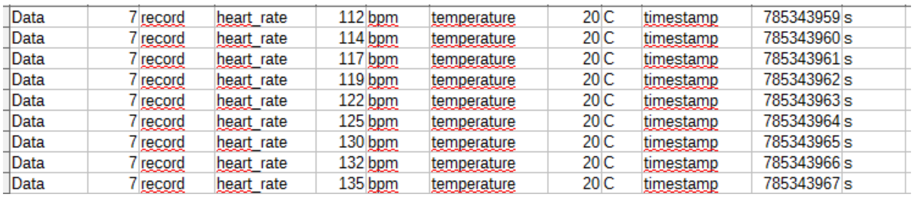
The data is stored in a FIT format with the name based on the format: $FIT\_YYMMDD\_HHMMSS.fit$, YYMMDD is the date and HHMMSS is the time. A FIT file that was created in January 1, year 2015 at the time 12:15:00 will get the name: $FIT\_20150101\_121500.fit$. The FIT file header is shown in the figure 7.4.



Type	Local Number	Message	Field 1	Value 1	Units 1	Field 2	Value 2
Definition	0	file_id	serial_number	1		time_created	1
Data	0	file_id	serial_number	3851629348		time_created	785343661
Definition	5	training_file	timestamp	1			
Data	5	training_file	timestamp	785343661			

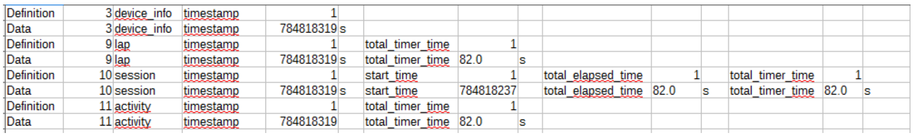
Figure 7.4: *FIT-file header.*

Once per second, data is written to the FIT file. How often data is to be stored depends on the type of application that is implemented. For training or situations where data must be saved at a higher frequency, it is important how often data is saved. A heart rate sensor sends data about 4 times per second while a environment sensor sends data every two seconds. For an application that for several days would just measure the temperature outdoors, it could be enough to save it every minute, or with a different time interval. At the end of the file information is stored about start time and total time of the activity as shown in figure 7.6.



Data	7	record	heart_rate	112	bpm	temperature	20	C	timestamp	785343959	s
Data	7	record	heart_rate	114	bpm	temperature	20	C	timestamp	785343960	s
Data	7	record	heart_rate	117	bpm	temperature	20	C	timestamp	785343961	s
Data	7	record	heart_rate	119	bpm	temperature	20	C	timestamp	785343962	s
Data	7	record	heart_rate	122	bpm	temperature	20	C	timestamp	785343963	s
Data	7	record	heart_rate	125	bpm	temperature	20	C	timestamp	785343964	s
Data	7	record	heart_rate	130	bpm	temperature	20	C	timestamp	785343965	s
Data	7	record	heart_rate	132	bpm	temperature	20	C	timestamp	785343966	s
Data	7	record	heart_rate	135	bpm	temperature	20	C	timestamp	785343967	s

Figure 7.5: *FIT-file recorded data.*



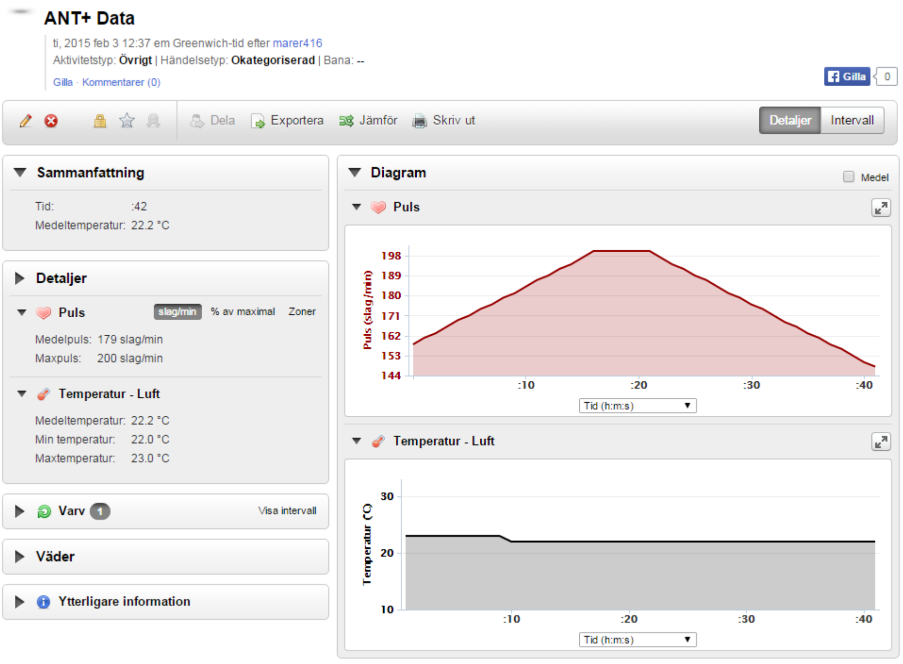
Definition	3	device_info	timestamp	1										
Data	3	device_info	timestamp	784818319	s									
Definition	9	lap	timestamp	1		total_timer_time	1							
Data	9	lap	timestamp	784818319	s	total_timer_time	82.0	s						
Definition	10	session	timestamp	1		start_time	1		total_elapsed_time	1		total_timer_time	1	
Data	10	session	timestamp	784818319	s	start_time	784818237		total_elapsed_time	82.0	s	total_timer_time	82.0	s
Definition	11	activity	timestamp	1		total_timer_time	1							
Data	11	activity	timestamp	784818319		total_timer_time	82.0	s						

Figure 7.6: *FIT-file closing lines.*

7.4.3 Verification

To verify the results of FIT files created there are different tools to use. In the FIT SDK provided by thisisant.com there are both examples of how FIT files should be structured and examples of how these might look like. FIT SDK also brings a converter for the FIT format to CSV, FITCVSTool. If the FIT-file that is created does not have the correct format or structure of content, an error message is displayed and the file will not be created. This is a perfect tool to ensure that the FIT file format is correctly created.

To be sure that the FIT file supports third-party manufacturers and applications the file is uploaded to Garmin Connect. Garmin Connect is Garmin's own website to store and display information about their training linked to their own products, where many of Garmin products support ANT + FIT and file format. In figure 7.7 so is a FIT file uploaded to Garmin Connect.



ANT+ Data

ti, 2015 feb 3 12:37 em Greenwich-tid efter marer416

Aktivitetstyp: **Övrigt** | Händelsetyp: **Okategoriserad** | Bana: --

Sammanfattning

Tid: :42
Medeltemperatur: 22.2 °C

Detaljer

Puls

Medelpuls: 179 slag/min
Maxpuls: 200 slag/min

Temperatur - Luft

Medeltemperatur: 22.2 °C
Min temperatur: 22.0 °C
Maxtemperatur: 23.0 °C

Figure 7.7: *FIT-file uploaded to Garmin Connect.*

7.4.4 Send files over ANT-FS

Sending files over ANT-FS has not been implemented. It took too long time to implement, and there was to little support for such an implementation in the ANT + Android API. More about that will be addressed in Chapter 8.

7.5 Evaluation

7.5.1 Methodology

Combining the Wenell project workflow (WPW) method with the agile development method in the Implementation phase has been a good way to work. The WPW gives a good workflow and focuses on what to do in each phase, to help achieve the goals and effects.

7.5.2 System design

Dividing the system into different layers has been advantageous as it can be more easily extended with other functionality, and gives an easier overview of the entire system. To extend the application with more features, like a mapping function, more sensors, or other functionality of the app, it's easy to get acquainted and understand where new parts of functionality are to be implemented.

7.5.3 Implementation

Using Scrum as approach for the implementation has worked well. It has been easy to use the requirements as a product backlog. To work according to an agile development method has been good because it has been easy to make changes. It has been valuable to be able to estimate time for different tasks, when to add more time or even terminate some sprints when the task goes in the wrong direction and requires too much work to be able to finish in time.

8 Discussion

The majority of the time has been spent on developing the Android application. Its been too big a task to build two applications, one for Android and one for Windows. So I should at the very beginning focused on just building one system, which had also been giving me more time on the Android application.

It is easy to extend the functionality for the Android application and to add more sensors. The same principle is used for all new sensors to implement with ANT +.

8.1 Send files over ANT-FS

One of the requirements for the application was to send files across the ANT-FS. After some investigation of the ANT + library, I realized that there is no support for ANT-FS. ANT-FS implementation would become too large to fit within the framework of this project, so I took the decision not to implement the ANT-FS.

8.2 Windows application

Much time was spent on trying to save the data in the Windows application. Since this took too long I choose to excluded it completely. In retrospect, I would have chosen to first make a small application that could only receive data and display it, not try to save away the data. The problem was that it was not as easy to store away data for Android application. The developing was very equal when ANT SDK is built on an equal basis for the different languages it supports (including both Java and C ++).

8.3 Android Application

I should have built the system so that it could easily change database. If in the future I want to add another or change database, the module can be replaced.

If more resources to the project were available the development of the application had been focused to put more time on functionality around how to save the data to that the users can choose how and where the data will be stored and handled.

8.4 Further work

Here are some comments on what you could be implemented in the future. This can continue to build on the functionality, both on the structure of the app and to add more features.

8.4.1 File format

At present, the user can not select the file format of the data to be stored. There are a lot of different file formats that manufacturers use. Some of those that seem most interesting to check further are:

- GPX (GPS Exchange Format)
- TCX (Training Center XML)
- HRM (Polar file format for heart rate)

These three formats are often used by different manufacturers in their applications. One function would also be able show data from previous practice sessions, import and export data to and from the application.

8.4.2 GPS

Many exercise and fitness applications support GPS, which is almost a requirement of users wanting to use applications for training and fitness. GPS with a map would be a fun and useful feature in the future.

8.4.3 Bluetooth

Since both ANT + and Bluetooth LE are large competitors, one could implements support of Blutooth LE in the application.

8.4.4 API/SDK

Both Apple and Google invest heavily in training and fitness industry. Apple released the application Health, and Google's app Google FIT. Google Fit SDK could be a tool to improve the application with more features and support for Google FIT. OpenfitAPI is another API that can be investigated for possible use.

8.4.5 ANT+

It is easy to extend the functionality of the Android application to add more sensors. The same principle is used for all new sensors to implement with ANT+. The problem with linking many sensors is how to visualize the data. In some cases the users may not have an interest in having the data visualized during a workout, but just seeing that information afterwards when analyzing their training.

Appendix

A

ANT+ Profiles

A.1 ANT+ Profiles

A.1.1 Receiver Channel Configuration

Parameter	**Value**	**Comment**
Channel type	Receive (0x00)	The heart rate sensor is a transmit channel device; therefore the display or storage device must be configured as the receiver.
Network key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for ANT+ devices.
Transmission Type	0 (for paring)	The transmission type must be set to 0 for a pairing search.
Device Type	120	The device type shall be set to 120 when searching to pair to an ANT+ heart rate monitor.
Device Number	1-65535, 0 for paring	Set the Device Number parameter to zero to allow wildcard matching. Once the device number is learned, the receiving device should remember the number for future searches.
Message Period	8070	Data is transmitted from the heart rate monitor every 8070/32768 seconds (approximately 4.06Hz)
Search Timeout	(Default = 30 seconds)	The default search timeout is set to 30 seconds in the ANT protocol.

Table A.1: *ANT+ Receiver Channel Configuration*

A.1.2 Transmitter Channel Configuration

Parameter	Value	Comment
Channel type	Transmit (0x10)	The heart rate sensor is a transmit channel device; therefore the display or storage device must be configured as transmitter.
Network key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for ANT+ devices.
Transmission Type	1	ANT+ devices follow the transmission type definition as outlined in the ANT protocol.
Device Type	120	The device shall transmit its device type as 120.
Device Number	1-65535	Set the Device Number parameter to zero to allow wildcard matching. Once the device number is learned, the receiving device should remember the number for future searches.
Message Period	8070	Data is transmitted from the heart rate monitor every 8070/32768 seconds (approximately 4.06Hz)

Table A.2: *ANT+ Transmitter Channel Configuration*

A.1.3 ANT+ Heart Rate Profile

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 Byte (MSB of Byte 0)	The transmitter must toggle this bit every 4 messages. The receiver may not interpret bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 Bits (7 LSB of Byte 0, mask 0x7F)	Data Page Number = 0 (0x00)	N/A	N/A
1	Reserved	3 Bytes	The transmitter shall set the value = 0xFF. The receiver shall not interpret this data at this time.	N/A	N/A
2	Reserved				
3	Reserved				
4	Heart Beat Event Time LSB	2 Bytes	Represents the time of the last valid heart beat event.	1/1024 (s)	64s
5	Heart Beat Event Time MSB				
6	Heart Beat Count	1 Byte	A single byte value which increments with each heart beat event.	N/A	255 counts
7	Computed Heart Rate	1 Byte	Instantaneous heart rate.	bpm	N/A

Table A.3: *ANT+ Heart Rate Profile*

A.1.4 ANT+ Environment Profile

Byte	Description	Length	Value	Units	Min/Max
0	Page Change Toggle	1 Byte	Data Page Number = 1 (0x01)	N/A	N/A
1	Reserved	1 Byte	Set to 0xFF	N/A	N/A
2	Event Count	1 Byte	increments with each measurement	N/A	N/A
3	24 Hour Low LSB	1.5 Bytes	Signed Integer representing the lowest temperature recorded over the last 24 hours (0x800 invalid)	0.1°C	-204,7 to 204,7
4 (bits 4:7)	24 Hour Low MSN				
4 (bits 0:3)	24 Hour Low LSN	1.5 Bytes	Signed Integer representing the lowest temperature recorded over the last 24 hours (0x800 invalid)	0.1°C	-204,7 to 204,7
5	24 Hour High MSB				
6	Current Temp LSB	2 Bytes	Signed Integer representing the current temperature (0x8000 invalid)	0.01°C	-327.67 to 327.67
7	Current Temp MSB				

Table A.4: *ANT+ Device Environment Profile*

Bibliography

[1] HUI Research AB. Christmas gift of the year 2014. 2014. Cited on page 2.

[2] Apple. Healthkit. 2014. Cited on page 2.

[3] Bluetooth. Bluetooth fast facts. 2014. Cited on page 20.

[4] Bluetooth. The low energy technology behind bluetooth smart. *Bluetooth SIG Inc*, ?(?), 2014. Cited on page 21.

[5] Sinem Coleri Ergen. Zigbee/ieee 802.15.4 summary. 2004. Cited on page 22.

[6] Google. Google fit, 2014. Cited on page 2.

[7] Google. Ui testing. 2014. Cited on page 6.

[8] Mohamed Habbal. Bluetooth low energy – assessment within a competing wireless world. *Wireless Congress 2012: Systems and Applications*, 2012. Cited on page 23.

[9] Dynastream Innovations Inc. Ant+ device profile enviroment. (1), 2012. Cited on page 8.

[10] Dynastream Innovations Inc. Flexible and interoperable data transfer protocol. (1), 2012. Cited on pages 26, 27, 28, and 29.

[11] Dynastream Innovations Inc. Ant / ant+ defined, 2014,. Cited on pages 7, 8, and 18.

[12] Dynastream Innovations Inc. Ant file share technology. 5(1), 2014. Cited on page 19.

[13] Dynastream Innovations Inc. Ant message protocol and usage. (5.1), 2014. Cited on pages xii, 10, 11, 12, 13, 18, and 26.

[14] Dynastream Innovations Inc. The facts. 2014. Cited on page 9.

[15] Dynastream Innovations Inc. This is ant. 2014. Cited on page 7.

[16] Jeff Sutherland Ken Schwaber. The scrum guide, 2013. Cited on page 4.

[17] Litepoint. Bluetooth low energy. 2012. Cited on pages 20 and 21.

[18] Daintree Networks. Applying mesh networking to wireless lighting control. 2004. Cited on page 22.

[19] Michael Owens. *The Definitive Guide to SQLite*. 2006. Cited on page 30.

[20] Farid Touati Rohan Tabish, Adel Ben Mnaouer and Abdulaziz M. Ghaleb. A comparative analysis of ble and 6lowpan for u-healthcare applications. 2012. Cited on page 23.

[21] Bill Schilit Roy Want and Dominik Laskowski. Bluetooth le finds its niche. *IEEE Pervasive Computing*, (1536), 2013. Cited on page 20.

[22] SQLite. Most widely deployed sql database. 2014. Cited on page 30.

[23] SQLite. Well-known users of sqlite. 2014. Cited on page 30.

[24] Dusan Stevanovic. Zigbee / ieee 802.15.4 standard. 2007. Cited on page 22.

[25] Wenell. Project workflow. 2015. Cited on page 3.

[26] Wikipedia. The scrum process. 2015. Cited on pages xii and 5.