Nikita Akmaikin

Sport-Oriented Hardware Development

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Electronics

Bachelor's Thesis

01.04.2016

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Nikita Akmaikin Sport-Oriented Hardware Development 22 pages 01.04.2016
Degree	Bachelor of Engineering
Degree Programme	Electronics
Instructor(s)	Thierry Baills, Senior Lecturer

The idea of this Thesis is making a research towards the development process of the open-source wearables platform. As most technologies at the moment do not use open-source principles solution is to see if the development can be finalized using open-source licensed devices with opened libraries.

This Thesis is covering the process development and reasoning of every component selection with explanation of programming principles, routing concepts and innovativeness up-to-date aspect. As the main role of the project is to build a platform that is usable for maximum amount of cases all the topics are covering multiple ways of solving one problem.

To test the system, sport application was selected to be the area under investigation. Theory covers the basic algorithms for Predictive Healthcare, Sport support and Sleep analytics at the same time explaining all the concepts of your heartbeat behaviour.

Main outcome of the project is to see if complex medical ideas can easily get implemented into the cheap open-source platform, providing same functionality and error/trial results as the branded solutions.

Keywords	Embedded programming, SPI, UART, ADC, I2C, MCU, STM32, Bootloader, Wearables, Open-Source, BeGo, Visual programming, Peripheral libraries, Communication principles.

Contents

Abbreviations

I2C – Inter Integrated Circuit

SPI – Serial Peripheral Interface

ADC – Analog to Digital Converter

UART – Universal Asynchronous Receiver - Transmitter

DMA – Direct Memory Access

TFT - Thin Film Transistor

LCD – Liquid Crystal Display

HR – Heart Rate

Ps – Pulse

VOM – Value of basal Metabolism

PAC – Physical Activity Coefficient

Kcal – Kilo-Calories

BPM – Beats per minute

1 Introduction

Today's market is showing a very big growth of wearable devices, which are updated roughly every day, changing functionality, design, material etc. However, once it gets to hardware development openness, wearables hardware availability and tools to work with, potential customer gets to a big trouble finding even one reliable solution. The understanding of this problem is very clear and straight - forward. To build innovations one always needs tools to build it.

During the past two years, market was facing only couple of changes, separating wearable devices in two types: smartwatches, sport trackers. The concept has been held so that development process was very different for both types and for sure, it is a problem. This project is an attempt to combine two possible wearable types into one environment, not only providing the design concepts but also showing the way to build a device with worldwide available components with opened libraries. Combination of the codes from community, hardware from the partners and innovations coming today this project proves the operability of the cheap platform implemented with complex sport ideas.

Platform runs on BeGo OS and is fully open-source. Generally, open source refers to a computer program in which the source code is available to the public for use and/or modification from its original design. Open-source is the main leading point of this project as sharing the code gives the opportunity to expand the amount of data received, solve problems and improve optimization.

Results clearly demonstrate that there is a possibility of building universal, low cost platform with functionality of setting use cases with any complex ideology.

2 Platform description

2.1 Hardware Overview.

2.1.1 Various principles in components usage.

Big amount of components changed during the development process. After couple of iterations, most important aspects were determined as the most influential:

1. Size of the component itself.
2. Price of the component.
3. Energy efficiency.
4. Cost of the board manufacturing using listed components.
5. Availability if the libraries for the listed components.

*Points listed in decreasing priority

By these principles, it is important to examine the market for the sizes of wearable devices available to calculate the proper size of the board. Branded devices were taken into consideration to make a proper comparison. Figure 1 presents the size comparison of selected devices. [2]

Figure 1: Size comparison of the branded smartwatches. [1]

Knowing the approximate sizes of the current models, size of the board was selected to be 43mm x 33mm x 0.6 mm allowing user to work with different screen dimension, battery sizes, components selection.

Considering dimensions mentioned above, components were selected.

2.1.2 Microcontroller - STM32F405

STM32F405 line is designed for medical, industrial and consumer applications where the high level of integration and performance, embedded memories and rich peripheral set inside packages as small as 4 x 4.2 mm are required. The first part of board design is to specify the minimal microcontroller schematics possible.

In case of the minimal platform there are 5 possible ways to limit the board size:
1. Bootloader type of loading
2. Powering and charging method simplification
3. Small peripherals packages
4. Buttons implementation in the borders
5. Routing should be kept simple and with a small width

While selecting the proper charging units there is always a need of selecting right powering methods. Decision towards micro-USB charging was made due to the method popularity and component availability with consideration of a low price range of components.
After our market research of components it was defined that most peripherals are functioning in range of 3.3 Volts while Microcontrollers consume 1.8-5.1 Volts, meaning 3.3 Volts solution ideally fits into the power standards. From the battery specifications, it is defined that Polymer batteries need a charging IC for providing 3.2-4.2 Volts, which gives an ideal 3.3 Volt output.
For a purpose of saving space, charging components were chosen to be in SOT23 package. Following units were selected:

LD3985 – 5 to 3.3 Voltage converter
MCP73831 – LiPo charger unit

Figure 2. Presents the powering method of the platform.

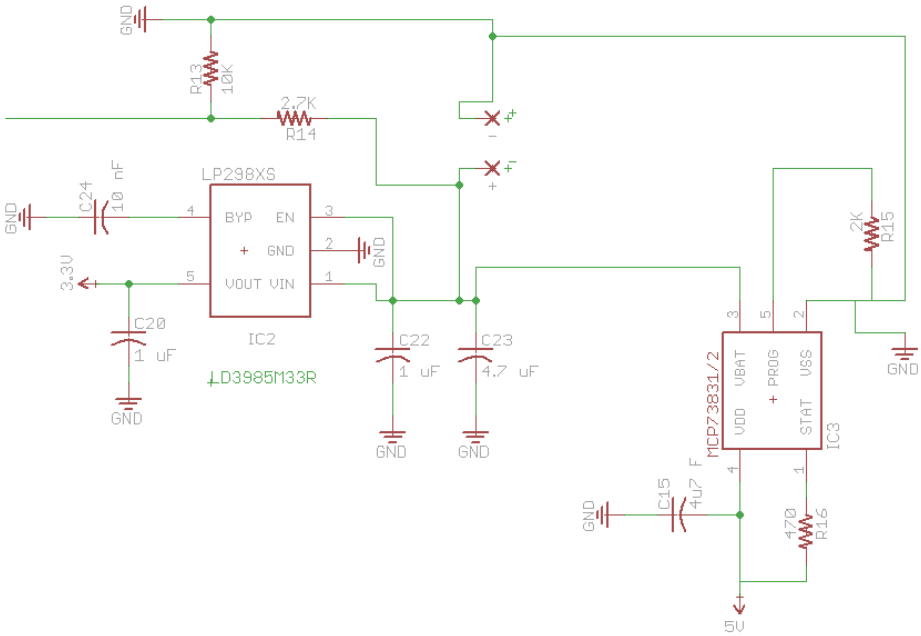

Figure 2: Powering method of the platform.

MCP73831(Battery charger) is provided with 5 Volts from Micro USB port and directed to SJ1 jumper that is intended for connecting plus and ground of the LiPo battery. At the same points, voltage is transferred to analog conversion through the voltage divider to provide system with accurate percentage of the battery. As it was mentioned before, most peripherals are powered with 3.3 volts, so amount less than 3.3 volts is considered to be 0% of the battery level.

Backup memory is still functioning even if the battery is 0% for providing Real Time clock counting and Wake up procedure.

LD3985 is a voltage converter with input interval of 3.3 – 7.8 volts, meaning MCP73831 is providing enough power for LD3985.

Another important role in powering the board is considering the external cabling. To follow current standards it was decided to use Micro USB input as the way to charge the device. Figure 3. shows the connectivity between input and the microcontroller.

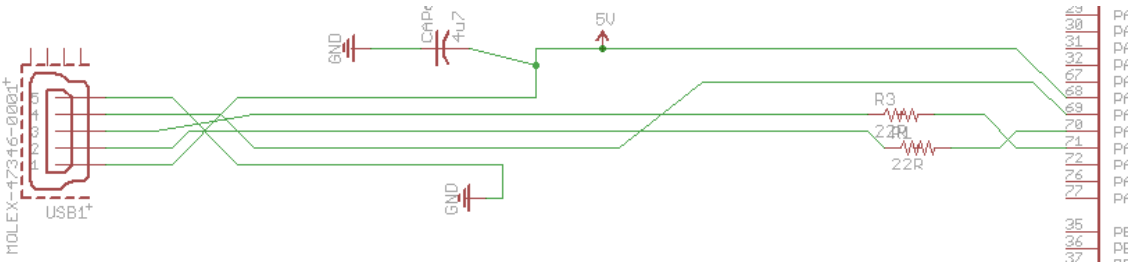

Figure 3: Micro USB connectivity.

2.1.3 Display – ST7735

Wide availability of displays on the market gives big opportunities in choosing screens between price tags, energy efficiency, drivers and communities. Considering development of the platform with limited sizes (43x33mm) there are certain dimensions that need to be reviewed. From that point of view, there are only couple of modules that provide almost equal sizes compared to the board. They are:

1. ST7735 display, 1.8 TFT
2. DSDM-160128, 1.8 OLED

Comparing two solutions, it was decided to use the first selection because of a price difference and available libraries, as we are following the open-source principle. One more advantage is related to amount of connections needed for screen operation, which is five connections straight to the microcontroller.

Figure 4. shows Display connectivity.

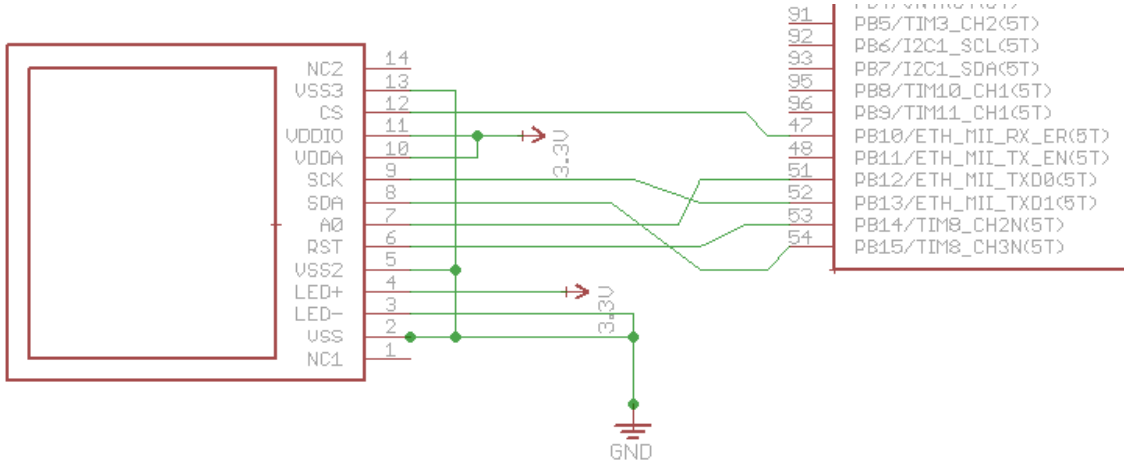

Figure 4: Minimal display connectivity.

For testing reasons, LED+ connection powered straight with 3.3 Volts. Ideally, this pin should be connected to microcontrollers pulse width modulation, which is able to provide proper 0-3.3 voltage interval. Reason for limiting this functionality now is an attempt to keep the idea of board universality to every possible project. For example, if someone is building a low energy solution powered with 1.6 volts then microcontroller is not able to provide a proper 3.3 volts, destroying the LED+ functionality.

2.1.4 Communication Unit - HCx Modules - Typology, Connectivity

Typology.

Selecting communication unit needs an understanding of possible future use cases for the board. There are many possible ways to connect different modules between each other. From the perspective of universality there are following selections possible [2]:

1. Xbee
2. Wi-Fi module (For example: ESP)
3. Bluetooth module
4. 2.4GHz communication unit

What makes these four solutions compatible with each other is equal powering and pinouts. All of them function using same connectivity with microcontroller. In case of development process, there is a need of connecting device to the smartphone as well as keep the device low power and provide possibility of allowing connection for old type of smartphones.

There are two Bluetooth modules reviewed in this Thesis:

1. HC-06 (Bluetooth 2.0)
2. HM10 (BLE 4.0)

First selection based on idea mainly covering not included mobile versions to the list of supported devices. After market research of the android devices functioning in the world we claimed that worldwide usage of smartphones that are using 4.1 android is around 40%. On the other hand 90% of wearable devices are working on 4.2> android smartphones. Therefore, to the variety of wearables for android 4.1 devices is enormously small. Figure 5 and 6 present two Bluetooth variants that can be used.

These are features of the module:
Default baud rate: 9600, 8, 1, n
Built in antenna
Coverage up to 30ft
Bluetooth version V2.0 + EDR
Operating voltage 3.3 volts
Item size 3.5 x 1.5 cm x 0.22 cm
Item weight 7g. [2]

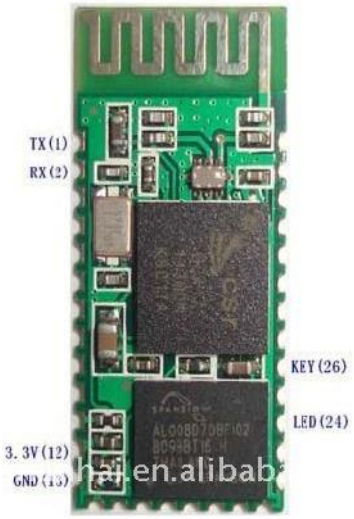

Figure 5: Bluetooth HC-06 module connectivity.

Default baud rate: 9600, 8, 1, n
Built in antenna
Coverage up to 30ft
Bluetooth version V4.0 BLE
Operating voltage 3.3 volts
Item size 3.5 x 1.5 cm x 0.22 cm
Item weight 7g.

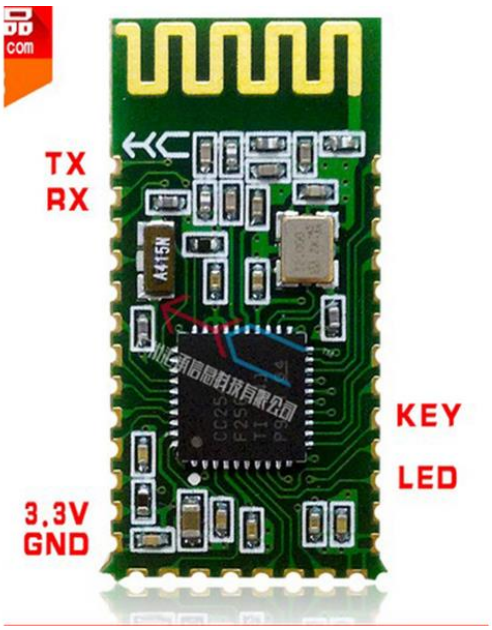

Figure 6: Bluetooth HM-10 module connectivity.

Based on modules configurations there is important aspect to mention. Both modules are interchangeable according to the datasheet provided. Meaning, based on the use case, there is a possibility of changing one to another only by soldering.
Considering that every day mobile devices are updated to newer firmware a need of better power solutions come up. HM10 compared to HC06 is more energy efficient and provides a faster wake up feature. BLE is intended for light duty cycle devices that support small data throughput and operate a long time on a coin-sized battery. Practical measurements showed following results:

Ongoing Bluetooth connection with paired device: 50ma consumption
Sleep mode with connection with paired device: 8ma consumption

Difference between these two conditions is a time needed for wakeup and receiving information.
By the means of microcontroller Bluetooth, connectivity is possible using Universal Asynchronous Receiver – Transmitter (UART). To have a proper transfer algorithm it is required to have a crosswise connection between Bluetooth module and Microcontroller UART pinout. Meaning connection is following:

Bluetooth TX (Transmit) – Microcontroller RX (Receive)
Bluetooth RX (Receive) – Microcontroller TX (Transmit)

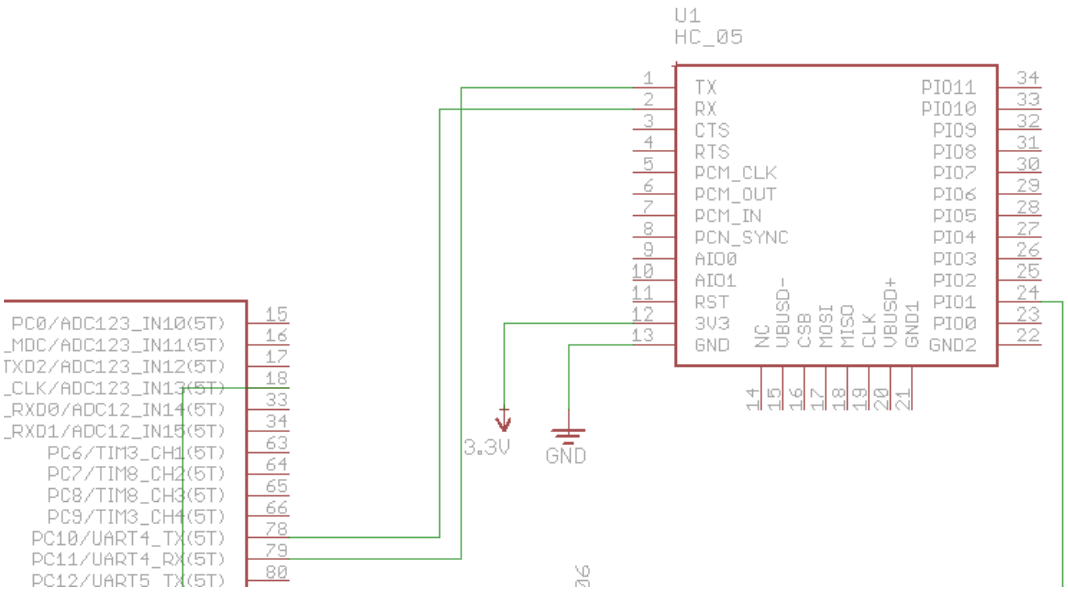

Figure 7: Bluetooth Module connection to the Microcontroller.

Figure 7. above shows the principle of connecting two devices with each other. The device symbol based on HC-05 module that is having same size specification as the modules mentioned before.

2.1.5 Gyroscope – MPU6050

As one of the possible applications is pedometer, it is highly important to use a proper accelerometer + gyroscope to get more accurate information. Following topics should be covered while choosing the right module:

1. Which accelerometer to use, analog or digital?
2. Accelerometer configuration, single or multiple axes?
3. What kind of sensitivity range is required, highly sensitive or middle? [3]

1. The choice between analog and digital is dictated by the hardware used. Analog accelerometers have an output that is a continuous voltage proportional to the acceleration. Digital accelerometers typically use pulse width modulation so that there is a square wave at a certain frequency. In this case, the time period of high voltage is proportional to the amount of acceleration.

While Stm32 Microcontroller in LQFP64 package has enough digital inputs it is to be the best to choose a digital output type of the accelerometer.

2. For most designs, two axes are sufficient. Naturally, a 3D project will require a 3-axis accelerometer, or possibly using two 2-axis accelerometers mounted at right angles. Most accelerometers are designed to sense movement in one direction, so 3D position sensing uses three crystals mounted in different orientations each with their own floating mass. For the most purposes of the platform, it is logical to detect a 3D motion. [3]

3. When measuring tilt using Earth's gravity, a ±1.5 g accelerometer will be more than sufficient regarding swing. If measuring the motion of a car or a robot, ±2 g is recommended. In addition, if the project involves sudden starts or stops, consider an accelerometer that handles ±5 g or more. Greater sensitivity in the accelerometer results in more accurate readings, so the more sensitive, the better. Sensitivity is the ratio of change in acceleration (input) to change in the output signal. This defines the ideal, straight-line relationship between acceleration and output. Sensitivity is specified at a particular supply voltage and is typically expressed in units of mV/g for analog-output accelerometers, LSB/g, or mg/LSB for digital-output accelerometers. It is usually specified in a range (min, typ., max) or as a typical figure and percent deviation. For analog-output sensors, sensitivity is radiometric to supply voltage; doubling the supply doubles the sensitivity. [3]

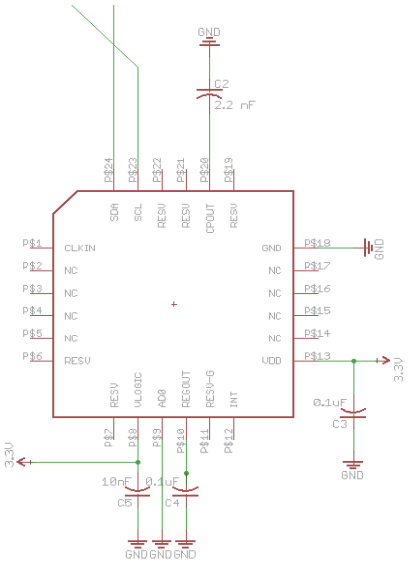

Figure 8: MPU6050 pinout.

Specifications:

Chip: MPU-6050
Power supply :3-5v
Communication mode: standard IIC communication protocol
Chip built-in 16bit AD converter, 16-bit data output
Gyroscopes range: ± 250 500 1000 2000 °/s
Acceleration range: ± 2 ± 4 ± 8 ± 16g

It is quite important to consider the pinout of the module as the amount of communication pins for I2C on the STM32 is limited. Following routing was chosen:

VCC,GND – powered by STM32 output with 3.3 volts
SCL – I2C Clock line Pin
SDA – I2C Data line Pin

2.1.6 Heartbeat Sensor - TCRT1010

Heartbeat sensors are reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light. There is a possibility of creating a wide functionality using the component, for example optoelectronic scanning and switching devices, index sensing, disc scanning etc.

For the use case of this project heartbeat application was chosen. Selection criterion is mostly same as for every component. It is price, size and energy efficiency. That was a reason for selecting TCRT1010.

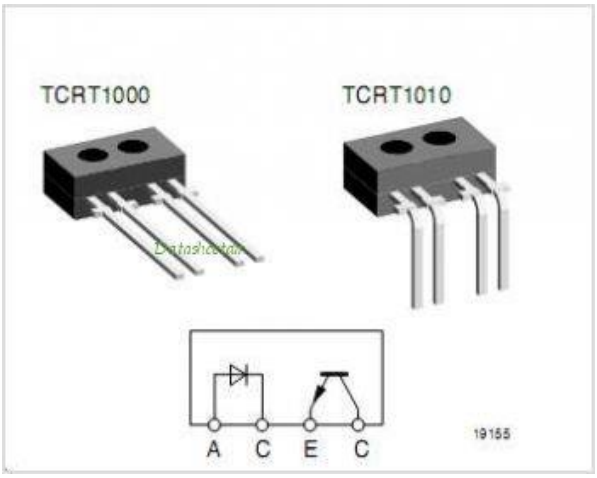

Figure 9: Simple diagram of the heartbeat sensor.

As it can be seen from Figure 9 the device itself is an analog output device. Next figure 10 shows the connectivity between device and microcontroller.

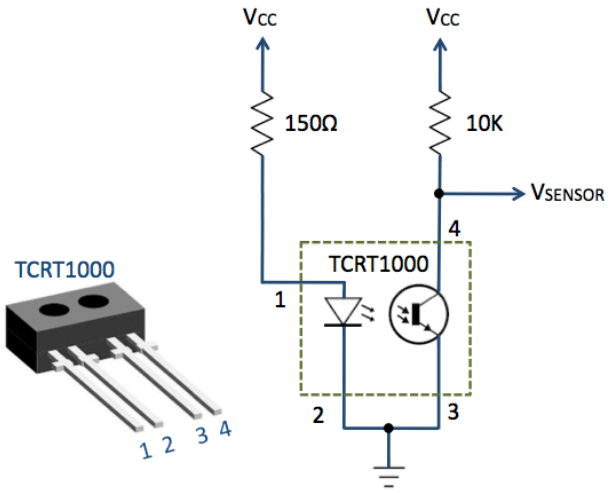

Figure 10: Simple diagram to connect heartbeat sensor to the board.

2.1.7 Loading Principle – Bootloader

This platform is following an open-source idea that is strongly dependent on programming opportunities for community. One of the ideas is to keep possibility of upgrading the platform, hardware of firmware as well as update different parts of the code. However, size and power consumption should still be in top of the priority list. Bootloader was chosen to be the loading method of the microcontroller.

The bootloader is stored in the internal boot ROM memory (system memory) of STM32 devices. It is programmed by ST company during production. Its main task is to download the application program to the internal Flash memory through one of the available serial peripherals (USART, CAN, USB, I2C, SPI, etc.). A communication protocol is defined for each serial interface, with a compatible command set and sequences. To ease the process of uploading a new firmware we are choosing a USB method to upload the code. [4]

Figure 11 shows micro USB pinout with Data pins for a transfer feature.

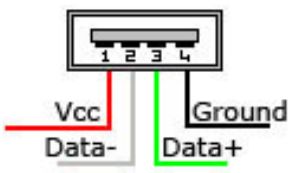

Figure 11: Data pinout for launching into bootloader mode.

Previously in Figure 3. it was described how the micro USB connection is processed. As it is important to keep the energy low we consider the idea of activation method for USB. By the datasheet provided from ST company following timing principles are determined: USB connection timing is the time that host should wait for between plugging the usb cable and establishing a correct connection with device. This timing includes enumeration and DFU components configuration. USB connection depends on the host.

Figure 12. Presents the idea of USB algorithm.

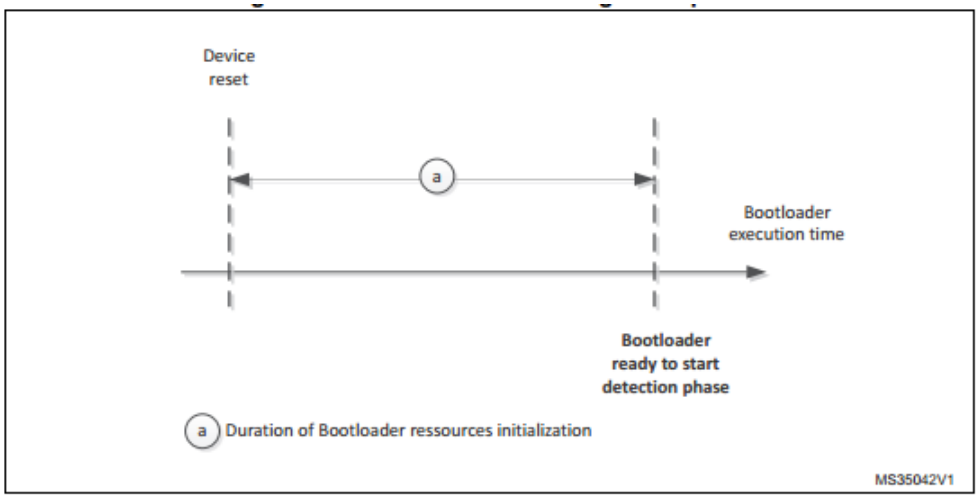

Figure 12: USB connection algorithm.

Duration of Bootloader resources initialization is provided in the datasheet and is showed in the Figure 13.

STM32F40xxx/41xxx	V3.x	270
	V9.x	250

Numbers are provided in ms.

Figure 13: Duration of Bootloader resources initialization.

To conclude theoretical part there are couple of outcomes:

1. Device needs a Reset procedure
2. After Reset pause needs to be processed
3. Bootloader phase initialization procedures finished the procedure. [4]

It was decided to choose a button method for accessing bootloader as stm32f4xx is provided with Reset and Bootload pins. That makes whole idea of connection easier. Looking at the graph on Figure 12 we can represent the idea as follows:

Reset button should be closed to get high on NRST pin for a duration of bootloader initialization and later bootloader button should be pressed to have high on Boot0 pin.

Figure 14. provides the button connection schematics:

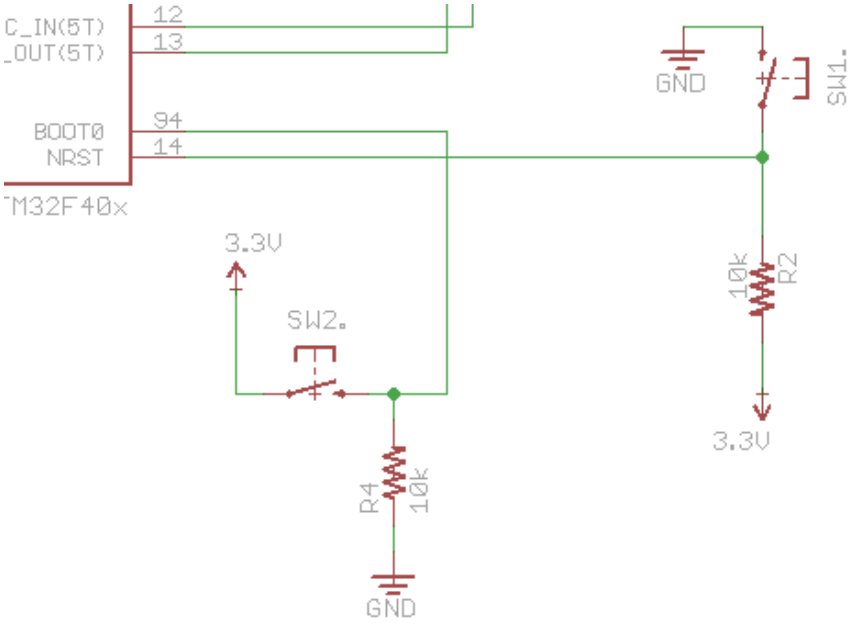

Figure 14: USB Bootloader buttons connection.

To finalize the procedure of accessing the boot process next actions should be kept in right order:

1. Holding reset button for 270ms
2. Pressing and holding bootloader button.
3. Releasing buttons.

As the system is using DFU mode for upgrading the firmware user need to have a .DFU file generated out of the code.

2.1.8 Overview of the board

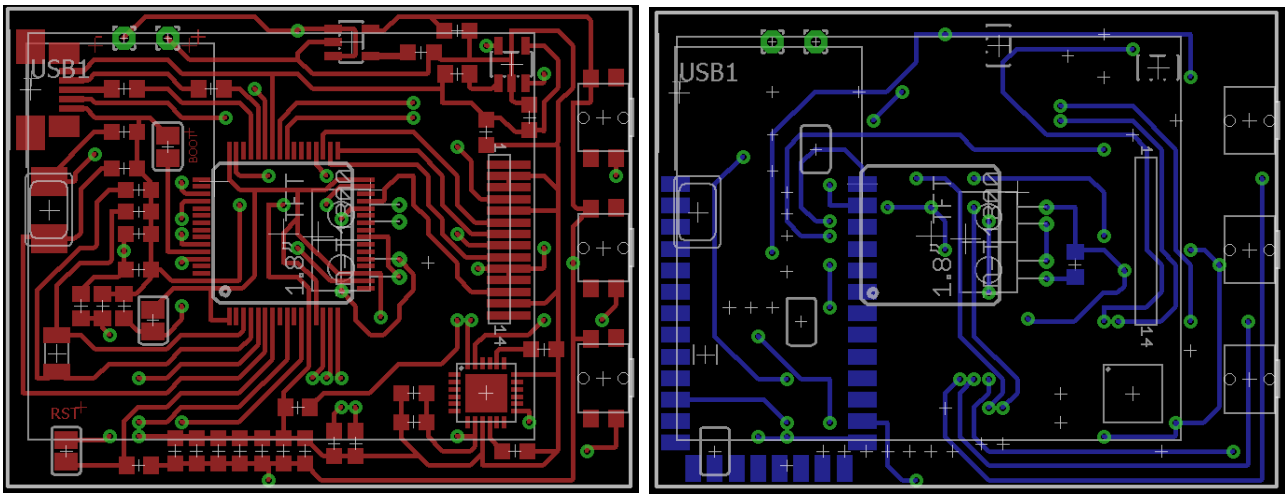

Figure 15: Top/Bottom layers

2.2 Hardware Initialization Algorithms.

2.2.1 UART Initialization.

UART is a universal synchronous receiver – transmitter (serial port). This peripheral has a raft of features for a huge range of serial protocols including all the usual modes plus IrDa, LIN, Smartcard Emulation and the ability to function as a SPI port. [5]

Our platform is using Bluetooth module meaning UART is chosen to be the communication principle as described above. While sending all the information character-by-character microcontroller needs to process the input. Figure 16 provides the input configurations for UART channel.

```
/* Configure USART Tx as alternate function  */
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC, &GPIO_InitStructure);

/* Configure USART Rx as alternate function  */
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

Figure 16. Inputs for UART channel.

After the input pins are initialized, it is recommended to set up the channel speeds and parameters of the module used. At this point universality aspect of the platform itself is proved, meaning what so ever communication unit is used, with the same protocol you are able to program the communication parameters per every different device.

```
USART_InitStructure.USART_BaudRate = 9600;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

USART_Init(USART3, &USART_InitStructure); /* USART configuration */
NVIC_EnableIRQ(USART3_IRQn); // enable Usart interrupt
USART_Cmd(USART3, ENABLE);/* Enable USART */
```

Figure 17. Setups for UART.

By the means of UART > Bluetooth connectivity methodology working setups should be same for those two modules. Bluetooth specifications are provided in Figures 6 and 5. [5]

After UART channel is configured, it is required to process the incoming/out coming data. For that purpose interruptions are used to handle the process.

```
void USART3_IRQHandler()
{
    if (USART_GetITStatus(USART3, USART_IT_RXNE) != RESET) // interrupt if something recieved on RX
    {

        USART_ClearITPendingBit(USART3, USART_IT_RXNE); // Clear Flag
        receivedData[receivedDataCounter]  = USART_ReceiveData(USART3); //recieved data equals to data from rx
       // USART_SendData(USART3,  sendbt); // echo of recieved data
        receivedDataCounter ++; // counter increase

        if  (receivedDataCounter == bytesToReceive) //data saved, clear buffer
        {
            USART_ITConfig(USART3, USART_IT_RXNE, DISABLE);
        }

    }
}
```

Figure 18. UART Handler.

To determine the idea of the code it is divided into 4 theoretical pieces:

1. If something is received flag in raised
2. Data variable is written
3. Counting amount of symbols received to write the array
4. Clearing buffer.

2.2.2 ADC Sampling

An ADC (Analog to Digital Converter) is a peripheral that allows measuring the voltage (between 0 – 3.3 volts in our case) on a certain input of the microcontroller and converting it into a number of the pulses per second, depending on ADC resolution. For example if 12 bit ADC is used the output result for 3.3 volts will be 4095 pulses. In case of the BeGo platform, there are couple of peripherals that use ADC conversion. They are: Battery capacity, Heartbeat sensor. As multiple peripherals are used, there could be a need for future memory access. For that purpose DMA is selected to access the memory.
DMA (Direct memory access) can do automated memory-to-memory data transfer, also do peripheral to memory and peripheral-to-peripheral. DMA channels can be assigned one of four priorities level: very high, high, medium, low. Moreover, if two same priority channels are requested at the same time – the lowest number channel gets priority. [6]

Figure 19 shows the ADC principle working in pair with DMA.

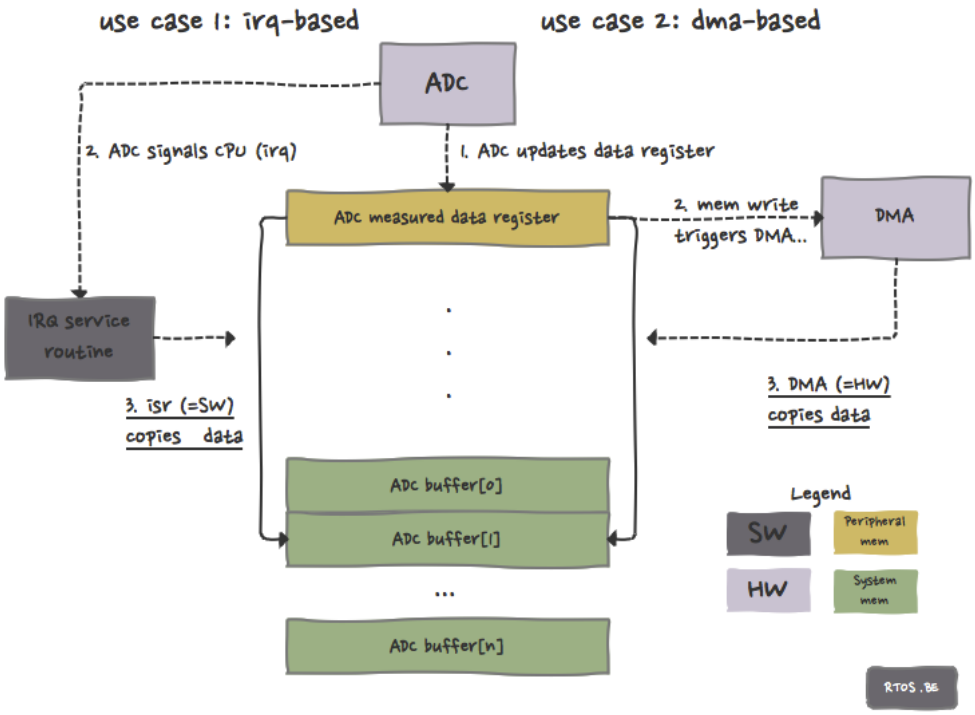

Figure 19. ADC and DMA relationship[6]

By the means of ADC, it cannot work by simultaneously reading all the analog ports connected to the board. Therefore, to work with three connections we need to read value

from we rather need to create interrupts for each of them or just use a Direct memory access – DMA.

Next all the DMA setups get determined:

```
/* DMA2_Stream0 channel0 configuration **************************************/
  DMA_DeInit(DMA2_Stream0);
  DMA_InitStructure.DMA_Channel = DMA_Channel_0;
  DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&ADC1->DR;
  DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)&ADCConvertedValues[0]; // First variable on ADC
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralToMemory; //Location assigned to peripheral register will be source
  DMA_InitStructure.DMA_BufferSize = 3; // amount of variables to write
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable; //automatic memory increment enable.
  DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;//source and destination data size word=32bit
  DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;//source and destination data size word=32bit
  DMA_InitStructure.DMA_Mode = DMA_Mode_Circular; //setting normal mode (non circular)
  DMA_InitStructure.DMA_Priority = DMA_Priority_High; // Priority setup
  DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
  DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
  DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
  DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
  DMA_Init(DMA2_Stream0, &DMA_InitStructure);
  /* DMA2_Stream0 enable */
  DMA_Cmd(DMA2_Stream0, ENABLE);
  /* DMA2_Stream0 channel0 configuration **************************************/
```

Figure 20. DMA Setup

This configurations are basically standard ones except buffer size. Also it is possible to decrease data size to save some memory. Now we continue with ADC post processing. Figure 21 describes the way ADC is initialized, separately from DMA initialization.

```
ADC_init_structure.ADC_DataAlign = ADC_DataAlign_Right;//data converted will be shifted to right
ADC_init_structure.ADC_Resolution = ADC_Resolution_12b;//Input voltage is converted into a 12bit number giving a maximum value of 4096
ADC_init_structure.ADC_ContinuousConvMode = ENABLE; //the conversion is continuous, the input data is converted more than once
ADC_init_structure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T1_CC1;// conversion is synchronous with TIM1 and CC1
ADC_init_structure.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;//no trigger for conversion
ADC_init_structure.ADC_NbrOfConversion = 3;//I think this one is clear :p
ADC_init_structure.ADC_ScanConvMode = ENABLE;//The scan is configured in one channel
ADC_Init(ADC1,&ADC_init_structure);//Initialize ADC with the previous configuration

ADC_CommonInitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_CommonInitStructure.ADC_Prescaler = ADC_Prescaler_Div2;
ADC_CommonInitStructure.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_5Cycles;
ADC_CommonInitStructure.ADC_DMAAccessMode = ADC_DMAAccessMode_Disabled;
ADC_CommonInit(&ADC_CommonInitStructure);

 ADC_RegularChannelConfig(ADC1,ADC_Channel_11,1,ADC_SampleTime_144Cycles);
 ADC_RegularChannelConfig(ADC1,ADC_Channel_12,2,ADC_SampleTime_144Cycles);
 ADC_RegularChannelConfig(ADC1,ADC_Channel_13,3,ADC_SampleTime_28Cycles);
 ADC_DMARequestAfterLastTransferCmd(ADC1, ENABLE);
 ADC_DMACmd(ADC1, ENABLE);
 ADC_Cmd(ADC1,ENABLE);
 ADC_SoftwareStartConv(ADC1);
```

Figure 21. ADC initialization.

3 Platform implementation into the sport application.

3.1 Correlation Between Heartbeat and Human Actions

At this point it is important to figure out the actions that can affect heartrate. For the research purposed there were selected Jogging and Sleeping states to make a totally different data variations.
First of all it is good to mention that HR depends on lots of factors like level of hormones in the blood which are synthesized on the morning, depending on human feelings, conditions, sleeping phase or even his nervous system.[7] Still, minimal HR appears during sleeping time, especially during the light sleep. Practically there could be a way to use that kind of information:
Building heartbeat norm of the person, comparing days, weeks, months. For example after collecting the data during the morning we are able to track the body fatigue, were there enough sleep during this night and even give an advice of sleeping more next night. After long term research there is a chance to predict a state of chronic stress.
It is important to mention that data acquisition is one of the influencing factors because HR of an adult can be 60-80 beats per second while marathoner can have 30-40 beats per second. So as conclusion it is a must to get a personal average data or a HR constant to know further critical changes.

Next step is to determine the HR factors for jogging. While running HR depends on many factors: sports type, fitness level of the body, sport type, exercise intensity, mental and emotional state, health condition, sex, age, temperature and humidity, etc.
There are maximum HR according to age and gender. For men, according to the formula:
For men: Ps max = 220- T (T=Age)
For women: Ps max = 226 – T (T=Age)
Depending on the fitness level of the athlete and the desired intensity of exercise, it is necessary to limit the percentage of increasing HR, which will contribute to the rational approach to training while gradually and gently increasing the running speed or endurance of the body.

Getting back to idea of using the BeGo platform there is a need to make a specific algorithm to use the information from above.

Theoretical approach can be divided into four parts:

1. Determine the maximum heart rate by the above formula by choosing gender and age of the watch user.
2. Before starting user need to choose the level: beginner, intermediate or professional.
3. Then choose distance: short or long.
4. The program makes the selection of variants of the maximum recommended heart rate:
0.6 * Ps max - if a beginner, or if it's a long distance (even for well-trained athlete) or for example there is an amendment that its hot weather outside.
0.7 * Ps max - if the intermediate level of training.
0.8 - factor for professionals on short or distances, if the temperature is average.
0.9 - the maximum recommended rate when training. For professional levels on short distances at normal environmental conditions.

During the run, program on the smartwatch will advise user to slow down, otherwise he will run out of breath and will not be able to make it to finish line. That is because there is not enough oxygen delivered to organs and the heart does not have time to saturate all the tissues and muscles. On other hand program can advise user to accelerate, as the heart still have more capacity.

These are ready coefficients. We can simply group them for different levels of training.

3.2 Theory Behind Heartbeat

The basic physiology of the body is the oxygen consumption of organs and tissues. During the physical activity, body shows increased demands for oxygen. There is a direct correlation between the intensity of physical activity and oxygen consumption of organs and tissues. If oxygen is provided at maximum, the body is able to cope with higher physical stress. If the required amount of oxygen cannot be delivered to the organs and tissues because of external (closed unventilated room) or internal reasons (violations of the cardiovascular system, lung diseases, diseases of the musculoskeletal system, exhaustion of the nervous system, or a large number of metabolism products in the muscles), a person will feel fatigue, weak, and a desire to stop work or training.

Heart rate is an indicator of the heart functioning, and increases when the heart needs to deliver blood with a portion of oxygen faster from lungs throughout the body to organs

and tissues (For example skeletal muscle). HR can also decrease when body has enough oxygen for normal functioning.

By looking through the factors: heart rate - is already known, the efficiency (physical exercise) - running speed + intensity of muscle work, IPC (maximum oxygen consumption of tissues of the body).

IPC - is the number of milliliters of oxygen consumed by the human body for 1 minute per kilogram of body weight.

HR - is directly proportional to the IPC and is inversely proportional to reflect the running efficiency.

Ps max (HR max.) ~ IPC

On average, the IPC of ordinary untrained people is about 40ml. At the highest qualification of athletes - 90-ml.Longer workout can be performed, with no more than 50% of the IPC. Greater percent of IPC, means less performance of the body. IPC and its percentage also depends on the intensity of workout. The greater the speed, the higher will be percentage of IPC.

Running efficiency can mean different things, dependingfor what purpose it is used. Efficiency can be used as running speed or the running duration (body's endurance).

3.3 Practical Application Of The Sports App Data.

Main purpose of the medical approach to the platform is to look for the sport oriented application from the different perspectives. It is very common to have a regular functionality like pedometer or heartbeat. Even advices about your jogging state becomes a standart for wearables these days. So using the concepts of the platform, especially open-source and public libraries allows the developer to use any references from the internet and implement it the the device easily. In case of sport oriented development our approach is called “Smart watch as a health guard”. Concluding the theory we come up with following:

Body is a dynamically complex system, in which the work of all organs and systems are in constant interaction. If all processes run smoothly, then we can talk about homeostasis (constancy of the system). As mentioned above, the heart rate is one of the indicators (quickly and easily available estimate) of homeostasis.

In some cases, heart rate can immediately increase as a reaction to certain processes, which take the body out of relaxed state. In this case, we can talk about such factors as physical activity, increased synthesis of thyroid hormones, stress and develop under the influence of adrenaline, activation of the sympathetic nervous system, the specific effects

of drugs and toxins, infectious agents, etc. Thus, a rapid increase in heart rate may indicate a specific effect of certain chemicals on the cardiovascular system or an acute process requiring increased oxygen consumption by the body. But it happens that increased heart rate is a sign that the heart cannot cope with its main function. Thats because its unable to deliver enough blood rich in oxygen to the tissues and organs by increasing power of contractions or increased volume of blood moved thru the artery. In this case, we can talk about the secondary increase in heart rate. This phenomenon is observed in diseases and pathologies of the heart and vascular diseases of the lungs, nervous system, organic pathology of the internal organs. Basically, it is typical for long flowing diseases. Very often person do not notice or does not think about his increased HR. Very small percent of people regularly monitors their heart rate. Meanwhile, there is a ot of factors causing this compensatory mechanism and we encouter them more than once on a daily basis. If the body is young, with well trained cardiovascular system and without chronic stress and chronic diseases – then short-term external factors will not harm health. On other hand if person has cardiosclerosis, for example, and neglecting any physical activity Will get in state of chronic stress by any reason.

In the last case, the high heart rate will indicate a serious decompensation of functions of vital organs and systems of the body. Therefore the state of body will agressively decline. Considering the smart watch and heart rate monitor function in dynamics, it must be said about the importance of them in the above question. Due to the initial individualization of standard parameters and the possibility to store index of heart rate for the past two months, a user will be able to understand when they should seek medical advice. This might help to prevent person on getting to doctor not when there are concrete signs of disturbances in body function or system, but on the initial phase of attempts of the body to normalize homeostasis, disturbed by external or internal factors. We do not say that the watch will be able to make a diagnosis. The device will help people to be more attentive to their health. This is important, because it is easier to prevent desease than to treat it.

3.4 Using Smart Devices for Health Improvement.

In the end of this research, we evaluate the final algorithm as a “Health guard”:

1. Creation of individual HR standards through systematic measurements in the morning after waking up every day in the first minute and then taking sample values for a week or two.

2. Monitoring HR and the growth of it, for example, which lasts more than 5-10 - minutes – therefore watch will give a warning on the screen with a signal if HR is increased. Then user can decide why it is so. (Jogging exercise is considered as a separate application, which have to be launched before starting your run).
3. If the heart rate level is not reducing to a normal rate (ex. for a few hours), the program will display a more serious warning.

3.5 Pedometer Actions Implementation

By theory pedometer is an electronic device that counts each step a person takes by detecting the person's hands or hips motion. Usually the mechanism that stands behind pedometer is rather an accelerometer, gyroscope or combination of both. In case of the BeGo platform, it uses the MPU6050 component, which was described on the pages above.

Every pedometer is different in functions. To make more reliable device there is a need of implementing most important features to decrease the error rate of every variable. BeGo platform covers following important features:

1. Distance estimate
2. Calories burned
3. Time management
4. Speed
5. PAC – Physical activity coefficient (Self calculated variable selected after research)
6. VOM – Value of basal Metabolism (Calculated variable selected after research)
7. Pulse monitor/helper
8. Daily goal

As most functions like Time management and Daily goals are more or less just programming background tasks, there are more significant concepts that explain the way of calculating amount of steps and distance done, at the same time showing a correct calorie burn. In the Figure 22. there is a clear view on the algorithm of calculating these variables.

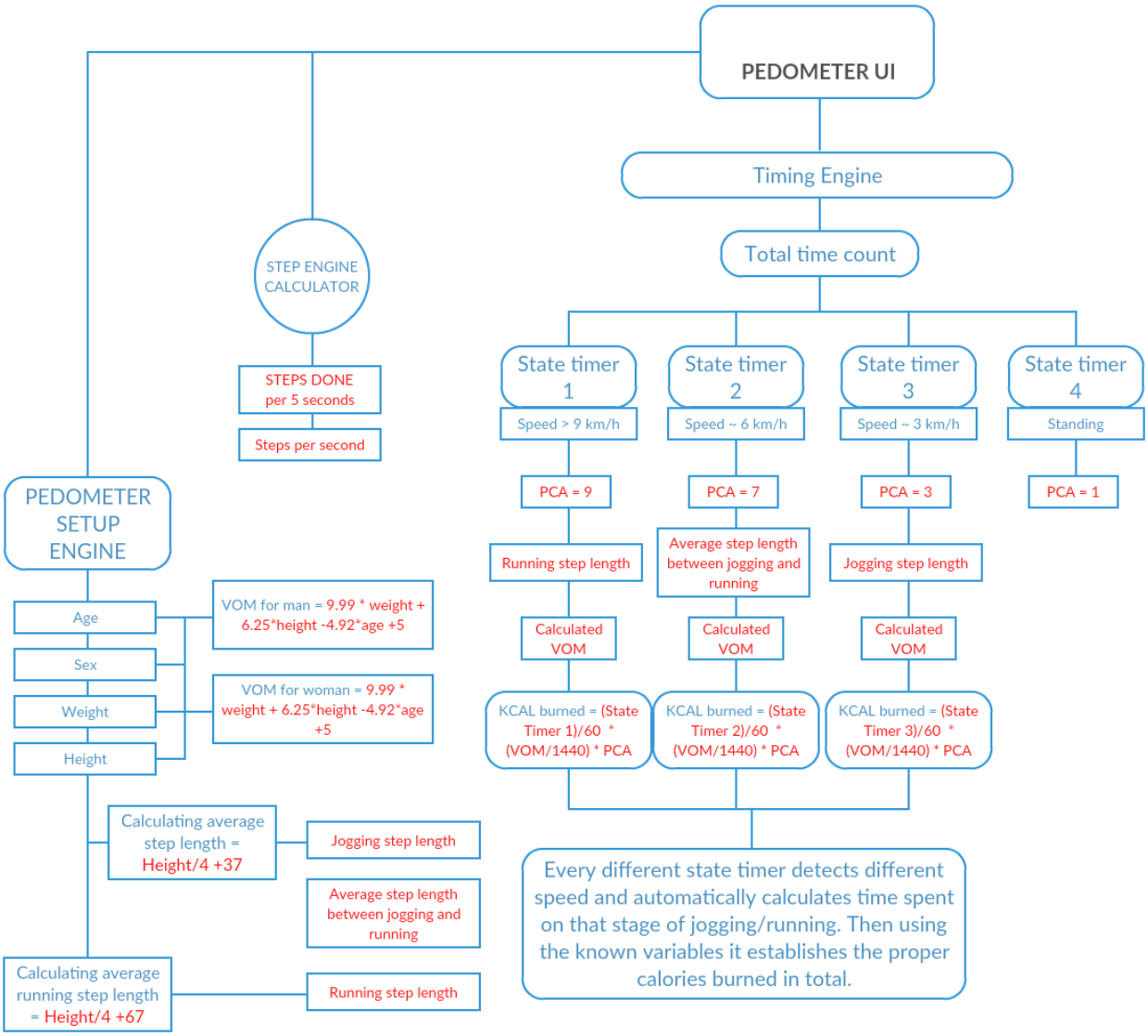

Figure 22. Pedometer algorithm used on the platform.

Theoretical explanation of the algorithm:

Basic principal of the system is that before launching the Pedometer application there is a need to go through Pedometer Setup Engine, which sets up your personal details (Age, Sex, Weight and Height). From those parameters, system is able to provide the VOM (The value of basal metabolism) for both women and men, also providing average step length of a person while running and walking.

On the background part there is always Step Engine Calculator running, providing amount of steps done per 5 seconds and per second. This engine works both in the application and on the background level of any task.

Another constant engine is Timing engine, which ticks the timer all the time, providing total time of the Pedometer service running. Important switching comes when the Pedometer Setup Engine is finished and all the data is forwarded to the timing management.

Total time count is dividing the tasking of your jogging into 4 parts. This is only limited functionality as ideally it should be unlimited amount of tasks at the same time. Later there will be explanation of the limitations of the system at the moment. Therefore, total time count is separating your running stages in following sequence:

1. First timer is switched on when the person is running on his maximum speed, which is more than 9km/h.
2. Second timer is switched on when the person is running on the middle speed, that is about 6 km/h
3. Third timer is working on the 3 km/h limit.
4. Last fourth timer is functioning when person is not walking at all, disabling all the timers.

All of those stages have pretty much the same variables inside every engine. All of them use VOM/PCA/step-length to finalize the results. One difference is that every stage has different values for these variables, which were calculated previously in the Pedometer setup engine. This separation of the stages allows user to have more precise data for every period of his jogging.

After the data is implemented into every different stage system automatically calculates how many Kcal were burned during the stage you are in right now. Once you get to another stage of running engine saves the previous data and starts calculating calories burn in another stage. User can never see on which stage he is right now. What is important here is that on the background level there is a Summation engine, which adds values from every separate stage into one buffer, showing a total Kcal burned, making no need for viewing separate values.

Example of this application can look like this:

Man is walking with 3 km/h speed for 10 seconds, after that he starts running with a speed of 10km/h for 50 seconds and then stops.

On application level it looks like that:

Assuming that person has made a Pedometer setup, then system knows that it is a man. Process is flowing so that firstly man is staying in State timer 3 summing Kcal every second and then after 10 seconds state engine switches State timer 1 remembering all

the data acquired in previous 10 seconds. Then it makes totally same calculations for 50 seconds period in State 3.
On the background level all this values are calculated into one buffer and displayed on the UI level of the watches. Example of GUI could be seen in Figure 23.

Figure 23. Possible GUI for pedometer.

There is possibility of extending the GUI level and for sure every different user wants to have different information displayed on the screen. This solution is something that was made after a 100 person questionnaire. But as it was described before Open-source is a powerful concept that has no limits in customization. Same here, as much as there is a need to display something more everybody are free to do that.
To conclude the algorithm part, once it gets to research of the possible pedometer algorithms there could be lots of similar solutions and at some point in open source world it is not even possible to find some really low-error, reliable code combined with hardware setup. So, what makes BeGo platform pedometer different from others?

From the perspective of calculating amount of steps done there is nothing special except detecting the hand movement and increasing the buffer with position values, at the same time ticking the timer to approximate amount of seconds spent in every position of your hand.

More relevant here is to talk about the middle part of the algorithm, explaining the way how to provide more correct data of calories burn, which is most important aspect when you are doing jogging. At this part there are two variables, which can improve the Kcal value.

First of all, in the Figure 22. it is clearly shown that separating part between Timing Count and calorie burn is VOM and PCA. During the research, we identified that combination as a unique one, which has never been used before. But what is it in theory and how were those values used in the algorithm.

PAC (Physical Activity Coefficient) – is the ratio of the average daily cost of human energy to the energy consumption at rest, so called value of basal metabolism. It is used to provide the person with information whether there is enough movements in his life or not. This coefficient provides the values for different activities, no matter is the person doing sports, sleeping, running or jumping. Every activity has a PAC level. It is also very important to follow the PAC level during human daily routine. Knowing that kind of information can help you to establish the proper physical activities during the day.

Average PAC level during the day is 1.75 and more.

In case of Pedometer algorithm uses four PAC values but in ideal picture it should be unlimited amount of values for every different speed while jogging. Reason for limiting that functionality at the moment is building the error percent between every stage. Fewer stages there are the easier is to build the correlation between speed and PAC.

To make it more clear in Figure 24. there is a table of PAC levels for future use.

Actions	PAC
Slowly walking	2,2
Jogging 3 km/h	3,3
Jogging 4.2 km/h	4,6
Jogging 5.3 km/h	5,8
Jogging 6 km/h	6,7
Jogging 7 km/h	7,8
Jogging 8 km/h	9
Running 8 km/h	8,8
Running 12 km/h	10
Running 15 km/h	13
Running 18 km/h	17
Running 20 km/h	40
Running 24 km/h	90
Sleeping	1

Figure 24. PAC values for different jogging speeds.

Another variable that provides importance for total Kcal burn is VOM.
VOM (The value of basal Metabolism) – the minimum number of calories needed to maintain the body's ability to live in a state of complete rest. VOM is the amount of energy (measured in calories), which the body will spend, if you sleep all day. Basal metabolism can burn up to 70% of the total number of calories expended, but this number changes depending on various factors.

VOM is determined by the following factors:

1. Genetics (not used in the algorithm)
2. Sex (used in the algorithm.) – For men VOM is bigger.
3. Weight (used in the algorithm) – Bigger your weight is bigger is VOM
4. Diet (not used in the algorithm, but still possible)
5. Body temperature (used as a constant, possible to make variable)
6. Outdoor temperature (not used, but exists in the platform)
7. Exercises (exists as a correlation between VOM and jogging)

By looking at this table, it is important to mention that at the moment platform uses only limited amount of VOM factors. Considering the use of every parameter results show that the information is more correct. In ideal perspective using more factors is better.

4 Conclusion

Idea of this project was to make a research towards the development process of the open-source wearables platform. Concept was to develop the board using affordable, easy to use, opened components to implement any possible case into the platform. For the purpose of this Thesis - Medical approach was used to show the possibility of using cheap system with complex computational algorithm.

On the beginning of the project it was decided to create a pedometer application with wide range of functionality that uses a pre-investigated values like VOM and PAC to correlate with jogging actions. This project is an attempt to change the way calories are calculated, providing the whole workflow of the project.

There is still plenty of space for improvements. This time it was not possible to prove the correlation between VOM and heartbeat. Despite of this, heartbeat values showed very great results in assisting personal activities, with possible future predictions and instant warnings about human conditions. Even providing the support for jogging, using heart-beat proves the possible ration between BPM and VOM.

Another significant part is the platform capacity. BeGo platform was 100% able to handle all the mathematical calculations, providing all the needed peripheral setups and showing great graphical opportunities.
This kind of project expands the borders of project development, showing that there are enormous possibilities in information sharing and information usage.

References

1. Will Shanklin. Smartwatch comparison Guide.[online]
   URL:http://www.gizmag.com/best-smartwatch-comparison-2015/40432/
   [Accessed 3 April 2016]

2. Luca Ruggeri. Top 5 Ways To Communicate With Your Controller. [online]
   URL:http://www.open-electronics.org/top-5-wireless-ways-to-communicate-with-your-controller/
   [Accessed 15 April 2016]

3. Carolyn Mathas. What You Need to know to choose an accelerometer. [online]
   URL:http://www.digikey.com/en/articles/techzone/2013/oct/what-you-need-to-know-to-choose-an-accelerometer/
   [Accessed 15 April 2016]

4. Magnus Unemyr. HOT TO DEVELOP AND DEBUG BOOTLOADER. [online]
   URL:http://blog.atollic.com/how-to-develop-and-debug-bootloader-application-systems-on-arm-cortex-m-devices
   [Accessed 15 April 2016]

5. No name. Basics of UART Communication. [online]
   URL:http://www.circuitbasics.com/basics-uart-communication/
   [Accessed 23 April 2016]

6. rtos.be . ADC driver implementation: DMA [online]
   URL:http://www.rtos.be/2013/03/adc-driver-for-the-energy-harvester/
   [Accessed 1 May 2016]

7. Malay Gandhi, Teresa Wang. The Future Of Biosensing Wearables. [online]
   URL:https://rockhealth.com/reports/the-future-of-biosensing-wearables/
   [Accessed 2 May 2016]