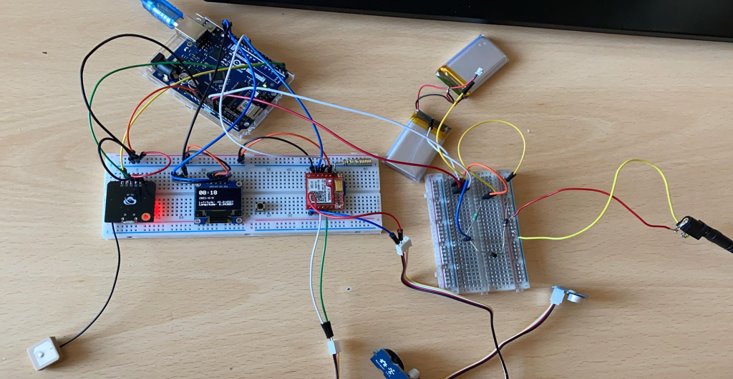
**Project Report**

*Topic: Smartwatch*

*Course: Connected System Design*

*Date : 15 June 2021*

**

*Team members*

*Aditya RANGANATH​*

*Alexandru MANDA​*

*Walid NAOUSS​*

*Joseph NILMARLAJ ​*

*Martin LE GOFF​*

**

**Table of contents**

Introduction​

I2C Communication and Project Structure

Charging Circuit

Sensor calibration

Project features​

* Fall detection​
* Geolocalisation​
* SOS message​
* Database connection and web interface​
* Pedometer
* Bluetooth APK connection​

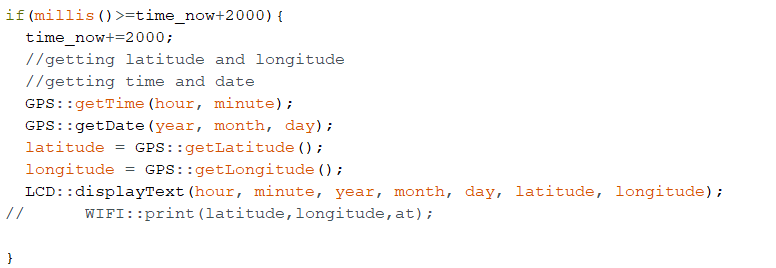
Conclusion​

**I2C Communication and Project Structure**

**Structure of the project**

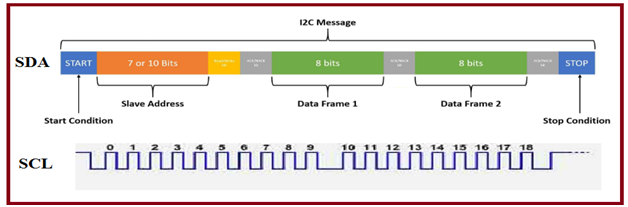
To structure the code so that all the modules are working in the same time, we have created separated .cpp and .hpp files for each module and different functions that will help in the main code to call each module during the setup or loop phase of the Arduino code.

The problem with the communication between the modules in the same time was that if we put a delay in a function of a module then all the modules will not have a proper communication since they will stop the communication with the serial monitor after a given time. The solution was to modify the delay function with different timers for each scenario. After every 1 minute we get the value for date time and GPS position and display it on the screen.



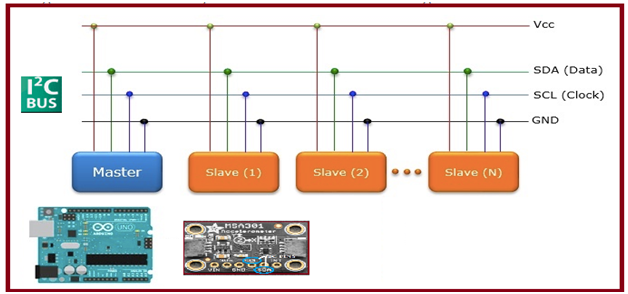
Also, an improvement will be an Arduino board with multiple hardware serial monitor ports, because the sim and the GPS use 2 different ports which he had to setup timers so that the communication will not be interfered. With Arduino Mega, the chip allows to have multiple communication between 4 parallel serial ports and have the possibility to create interrupts scenarios which are used in all the computers nowadays as a hardware architecture of the system.

**I2C Communication**



SCL and SDA as signal lines. Both lines are connected to VDDIO externally via pull-up resistors so that they are pulled high when the bus is free. At start condition, SCL is high and SDA has a falling edge. Then the slave address is sent. After the 7 address bits, the direction control bit R/W selects the read or write operation. When a slave device recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle.

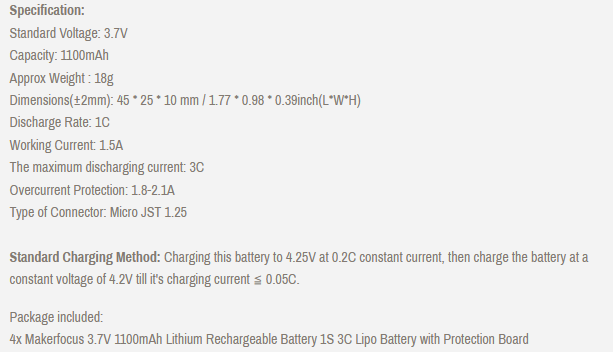
At stop condition, SCL is also high, but SDA has a rising edge. Data must be held stable at SDA when SCL is high. Data can change value at SDA only when SCL is low.



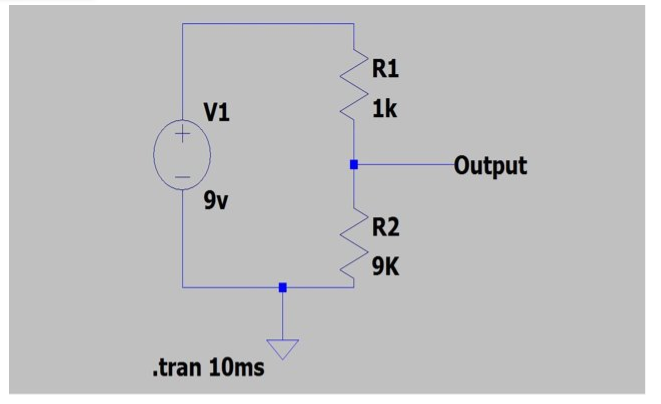
Also, for the communication part, if we are using multiple SDA and SCL devices, we can connect the devices on the same line and specify the address of each component (0x3C is the address for the LCD).

For this we have used also an Arduino program to scan the I2C devices available on the ports to know which is the address.

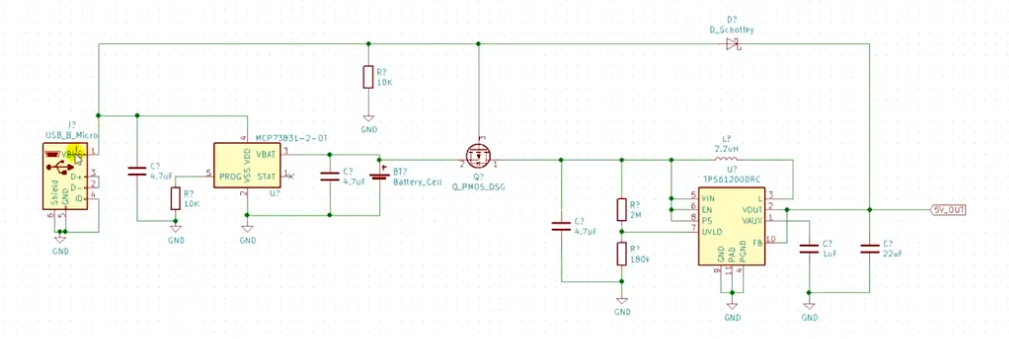
**Charging circuit**



For the charging circuit we used a voltage divider to drop the voltage from 9.16V (9v 2A adapter provided from lab) to 8.06 approximately. According to the specifications, every cell has the standard voltage of 3.7V, but even when the battery was provided to us the current measured on the multimeter was approximately 4V. And according to specifications, we need to charge every cell to 4.20V constant voltage until the charging current is <= 0.05C which means (0.05\*1100mAh = 55mAh) then we have to stop the charging. According to this, we were able to provide to the batteries a voltage of 8V, because if we have 2 cells we need to charge them with 4.2V \*2 = 8.4V until the current is 55mAh. Unfortunately, we were able to charge it with 50mAh constant current.



According to this, we have measured the requirements for current to power the Arduino and all the required modules, and we need a current of 100mAh and the charging will provide us around 50mAh. So when we try to charge the batteries and use the Arduino in the same time it will not charge, because of the current limitations of the voltage divider. However, we wanted to implement the charging circuit with the transistor, or with a charging chip module.



With this method, we will separate the charging and the voltage that the board and the modules need to power the Arduino, so it means that the charging circuit and the power required for the board will be two separate circuits.

**Sensor Calibration**

**Why sensor is needed to find Fall detection?**

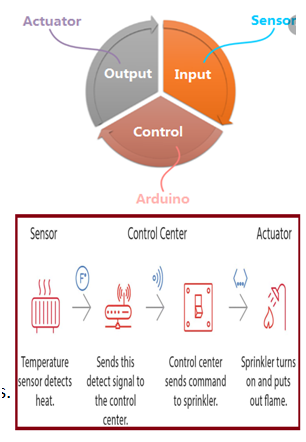
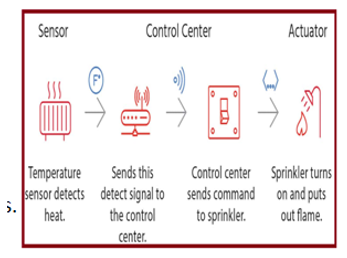
If a person falls from one rest position to another position, how do we detect? Yes, it’s possible by

* Eye contact,
* Physical touch and Noise

All these three points stated or descriptive in “Manual Measurement”, it leads more into slothful moments nowadays. Advantages of Utilization the technological aspects come into the picture,   
  
**Role of Sensor and its Potential**

How do we do detect when there is no one is placed?

* Role of sensor here is, in a control & automation system is to detect &measure some physical effect, providing this information to the control system.

Information in terms of electrical signal or non-electrical signals (Mechanical Further sensor information's are embed with micro controllers and transmit to the required user for the appropriate decisions.

**Sensor Accelerometer**

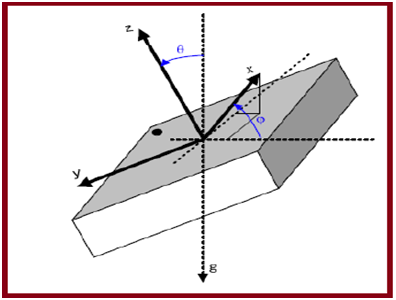
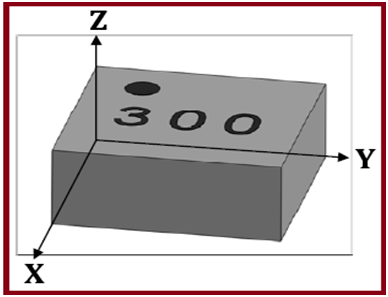
**Why accelerometer** and no other device like vibrating sensor or distance measurement sensor?

* Accelerometer include linearity over a wide frequency range and a large dynamic range and additionally it has capable of tri-axial measurement and I2C support.

|  |  |
| --- | --- |
| **Slno** | **Main Feature Content of MSA301** |
| 1 | Dynamical user selectable full scales range of ±2g/±4g/±8g/±16g |
| 2 | Allows acceleration measurements with output data rates from 1Hz to 500Hz. |
| 3 | Minimum supply voltage 1.62V to 3.6V and Maximum value -0.3 to 3.6V |
| 4 | Operating or maximum voltage of IO supply voltage is 1.2V to 3.6V |

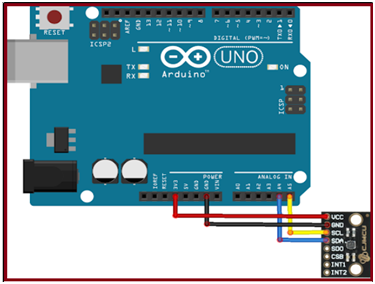
**Implementation of Sensor Accelerometer into our project**

1. The Orientation recognition feature informs on an orientation change of sensor with respect to the gravitation field vector ‘g’.
2. The accelerometer scale of each depends on the sensitivity settings chosen which can be one of +/- 2, 4, 8, or 16g for the accelerometer. The accelerometer produces data in units of acceleration (distance over time2).
3. The output scale for any setting is [-32768, +32767] for each of the six axes. Default of I2C is is +/- 2g for the acceleration. If the device is perfectly level and not moving, then:
4. X/Y accel axes should read 0 or less constant value
5. Z accel axis should read 1g, which is +16384 at a sensitivity of 2gwhich outputs the raw readings as mg/LSB.
6. **Output format**: mg = milli-G's, 1mG = 0.001 G's of acceleration, so 1000mG = 1G. And LSB = Least Significant bit, which is the last bit on the right. The raw values from the accelerometer are multiplied by the sensitive level.
7. The acceleration output is 14-bits two’s complement data. data for each axis is split into MSB part (one byte containing bits 13 to 6) and LSB lower part (one byte containing bits 5 to 0). To enable ‘NEW\_DATA\_INT’ function, (0x17) ‘DATA\_INT\_EN’ should be set.



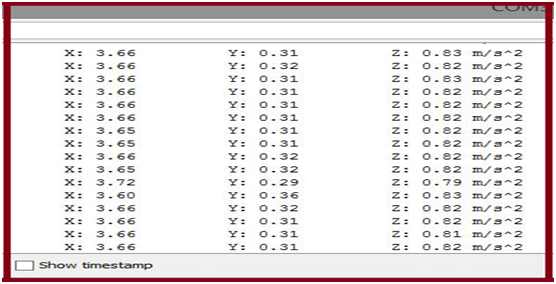
**Block Diagram**

Communication line between Sensor and controller by two wire connection SDA (data) & SCL (clock). Supply by Vcc and GND> Total 4 wire connections (3.3v or 5v would work).

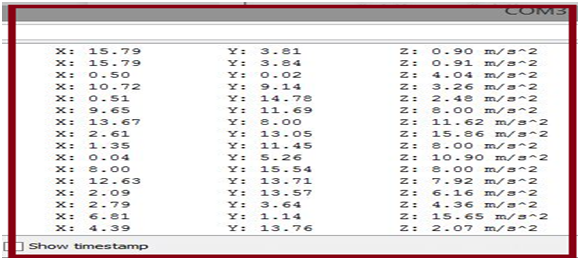


**Output from the sensor**

1. Rest position of the acceleration (orientation is stable and received constant value)



1. Changes in displacement (freefall with more than 30deg angle (Non Orientation))

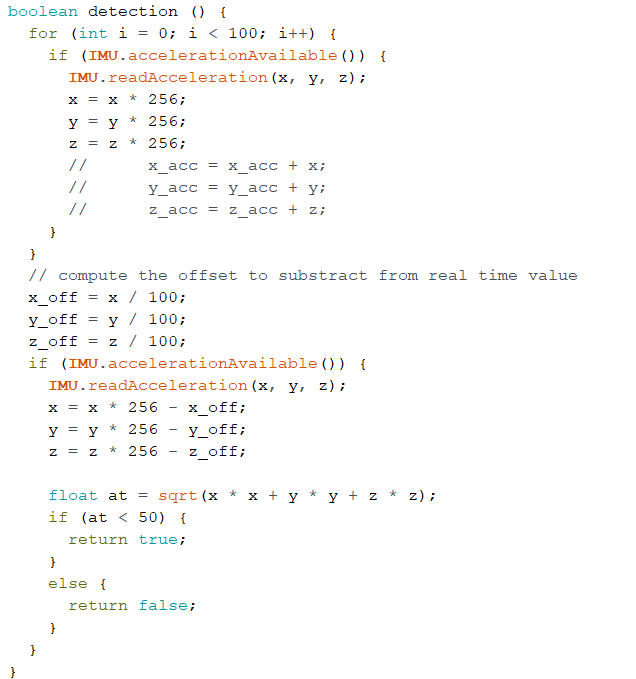


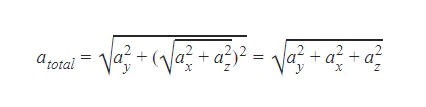
**Project Features**

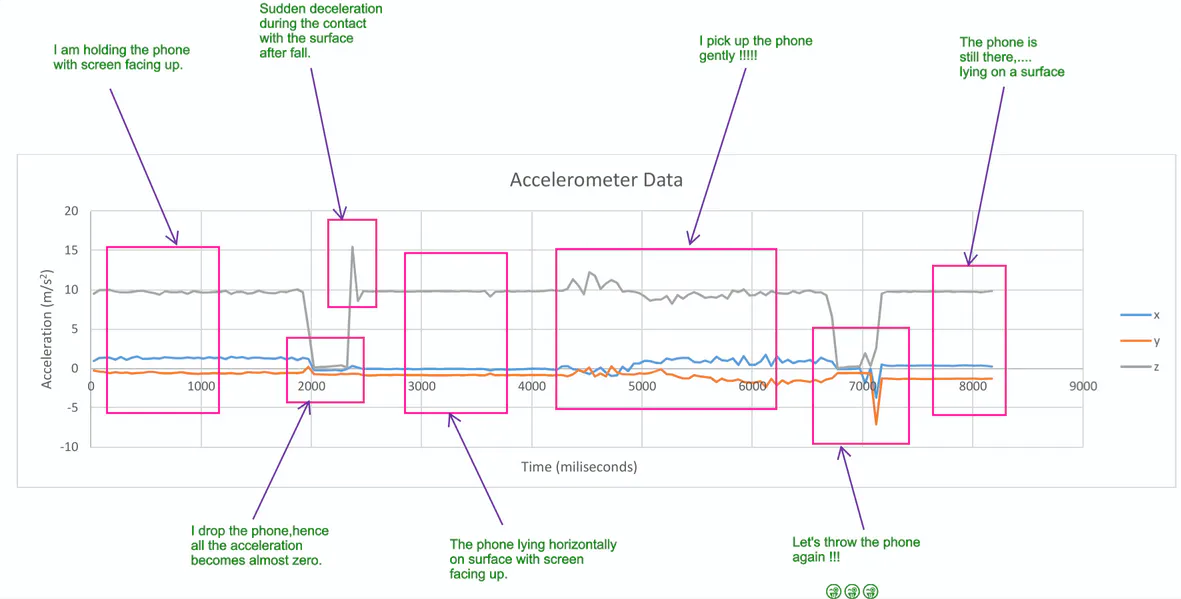
* **Fall detection**

Fall detection devices automatically employ the technology to detect and get fast assistance for a senior that is prone to falls.​

Fall detection systems use accelerometers, a type of low power radio wave technology sensor, to monitor the movements of the user. State-of-the-art fall detection devices use three axis accelerometers. ​

**** To calibrate the built in accelerometer available from the Arduino LSM6DS3TR inertial measurement chip we have choosed to multiply the value of the acceleration from each x,y,z axis with the 256 values to have a range from 0 to 1023 and then we compute the offset by dividing every axis offset and after that we substract the offset.

****The algorithm for fall detection is based on the total acceleration of every 3 axis which is the squared root of every axis ^2, and when the total acceleration reaches a certain threesold (50 in our case given the acceleration values range) then it detects the fall.

****

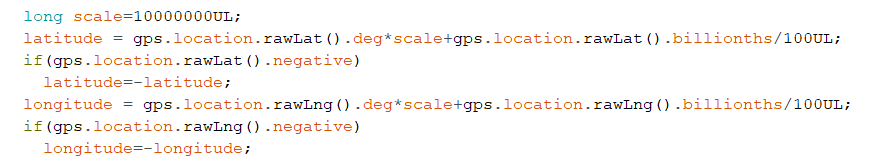
At beginning the phone is with screen facing up, almost horizontally and minor fluctuations in acceleration due to shaking of hands. As you can see the acceleration about z-axis is ~ 9.8 m/s^2 - the acceleration due to gravity. Then I drop the phone, the acceleration goes to zero about all three axis for a short time. Then at the end of the free fall the phone strikes the surface and feels sudden deceleration and comes to rest, evident from the spike. Note that this spike can occur about any axis depending upon the orientation of your phone at the instance of strike. After the strike to surface the phone is resting steadily, so the acceleration lines are straight horizontal suggesting the absence of even the minor fluctuations. Then I pick up the phone and repeat the procedure for one more time.

So. it is for sure from the above data is that the acceleration about all 3 axes drops close to zero during free fall. In our case, after the total acceleration computing, we can notice this phenomena when we try to drop the Arduino board the total acceleration value is decreasing due to spike of the acceleration value in one of the axis depending of how the Arduino is oriented.

We have tested our algorithm and it recognizes the fall if the total acceleration is bellow 50, which means that from a range of 0-1023 of values the total magnitude of the acceleration is dropping and then we can detect the fall.

* **Geolocalisation**

We have used the GPS module to get the information about time, date and latitude, longitude. However, during to Arduino limited memory storing, float numbers which are greater than 6 decimals can not be stored, so we have used the TinyGPS++ module to encode the GPS information in a nicer way and then we have stored it in a string with the precise location of 7 digits coming from the GPS module with the respective latitude and longitude. Also we have found a method to convert the raw values of the GPS module in a way that we will have a precise location with +/- 1.5cm error of measurement, which allowed us to calibrate the position issues given from the fact that Arduino has limited floating point arithmetic capabilities.

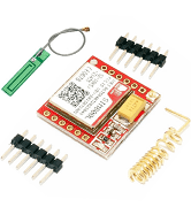
****

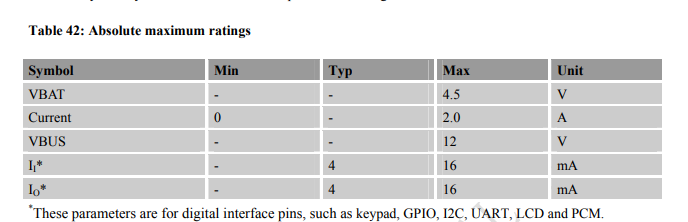
We have also discovered that the GPS GT-U7 module has a PPS pin which can be taken into consideration when we want to synchronize the time very efficiently with an accuracy up to 1nS. It can be very useful for syncing the RTC clock with the PPS pin giving accurate timing for interrupt processes.

Although we have discovered that we were not able to implement it in our project due to time constraints.



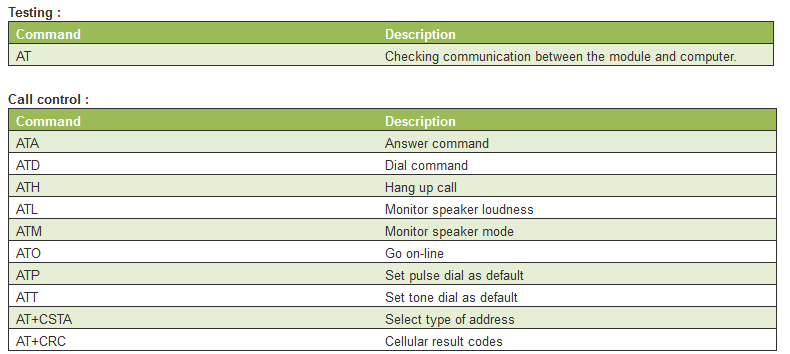
* **SOS message**

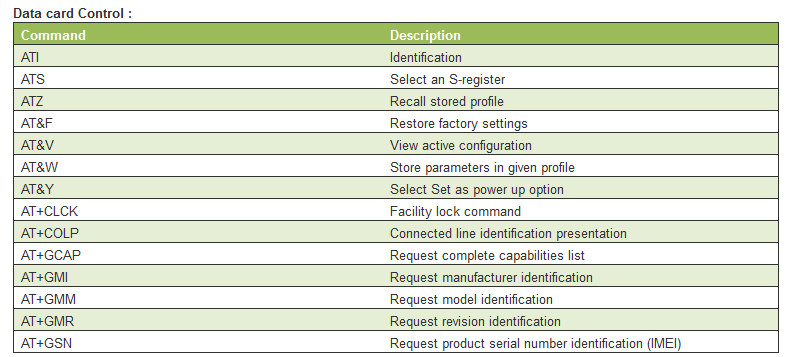
After we get the fall detection value (true or false) we have to understand how the SIM800L module functions and what are the requirements for that. First we needed to solder the helicoidal antenna for receiving the GSM signal and to be able to implement it on the breadboard.

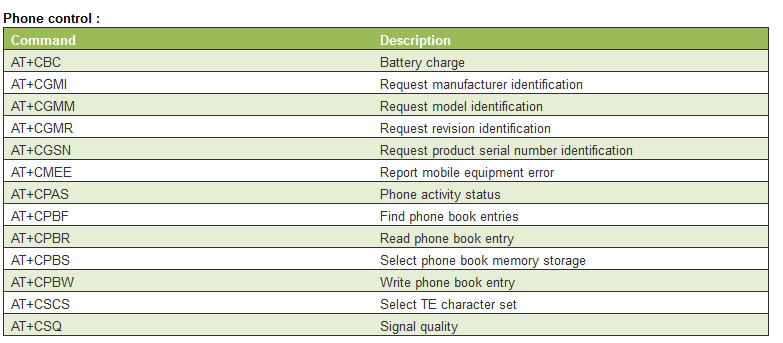
****

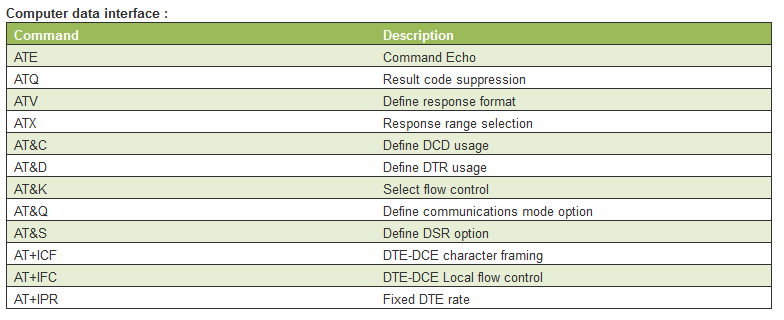
Although in the specifications the maximum voltage for this module is supported for 7V, but the recommended supply conditions are 4.5V and 2.0A current. So when we try to test with the USB communication from the laptop to power the Arduino, it will not connect to the network, because it doesn’t have enough current to connect to the network, but the moment we connect the battery we receive the signal and we are able to make calls, receive or send SMS, see the battery voltage status and many more features. Also this module comes with an RTC feature for Real Time Clock synchronization.

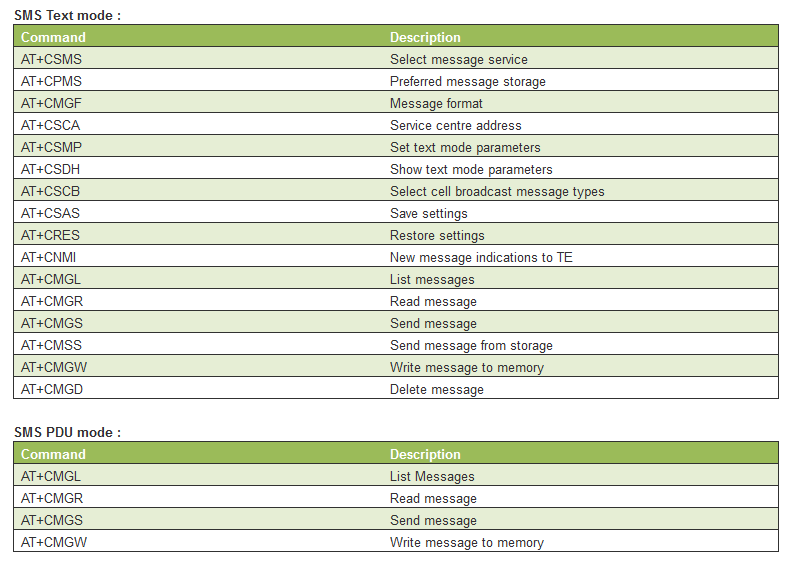
To be able to communicate with the sim modules we have to understand the different AT commands for call, SMS, battery pin, volume of the speaker and the mic and many more commands.

****

****

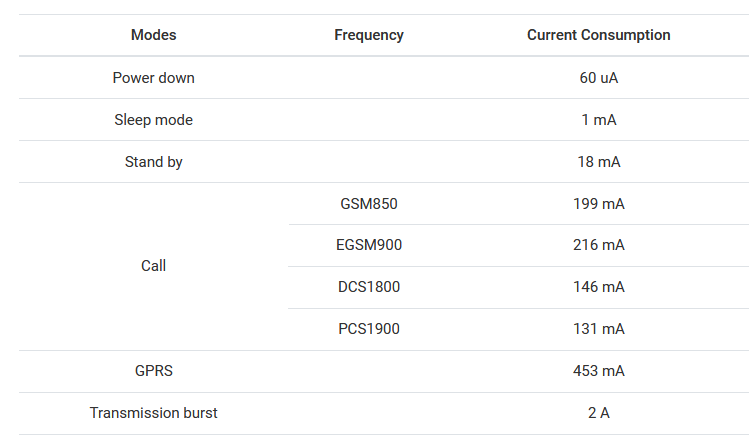
****

****

****



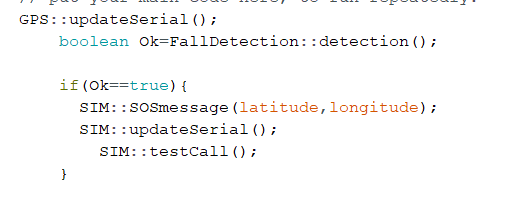
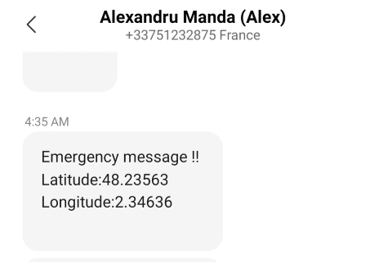
If the network status indicator blinks every 1s it means that it searches for network and if it blinks for 3 seconds it means that it is connected and ready to use the AT commands to send SMS, receive calls, send calls and so on.

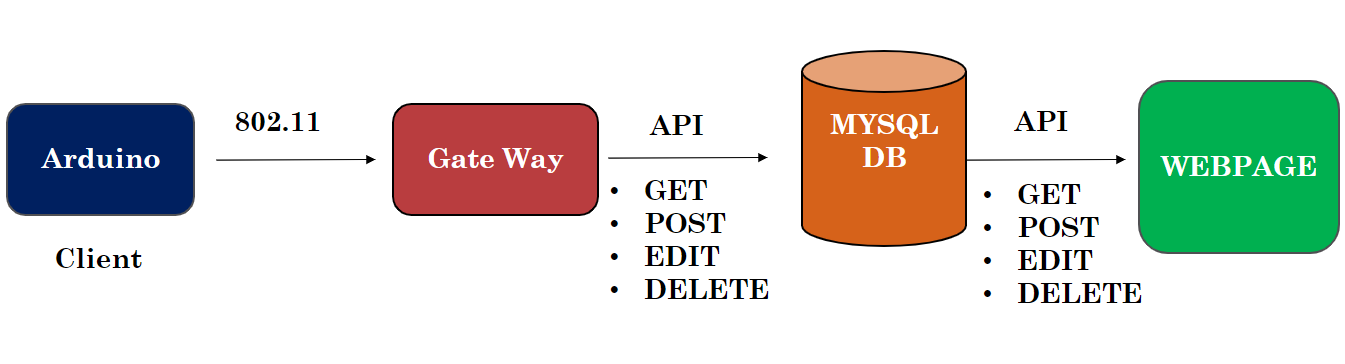
****

In the image above we can see what are the requirements regarding to the current consumption for every state of the module.

Also an important step is choosing the right antenna for the communication. We can solder the helical antenna to the NET pin to save space, but sometimes it struggles to connect if we are in an indoor situation. From our tests it works perfectly, but sometimes it needs some time to connect to the network to be able to make calls/SMS (1min approx).

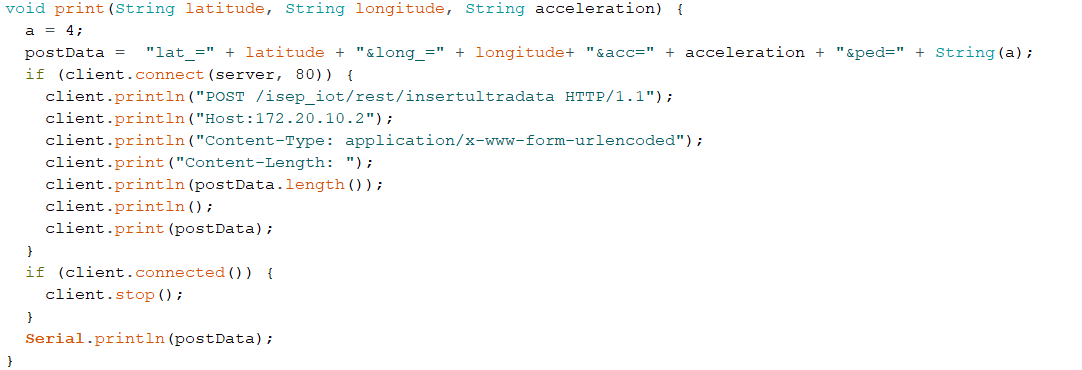
So after the fall detection we will send an SMS with latitude longitude information and a call to a predefined emergency contact to be able to communicate with the seniors in case of an emergency and also know the exact location in case we detect a fall and they don’t communicate through the automatic call made.

****

* **Website interface and database connection**

In our project we have created a web interface to store the values into a database for different parameters (latitude, longitude, acceleration magnitude and step counter). We have created Restful Web API services for the backend and the frontend is made with Angular Js. The idea is to store the values from the Arduino every specific time (for demo purposes we have tested every 1s to upload the data) to keep in track the date and time along with the location, and different parameters. The next improvement for the frontend part is to create different visualizations such as charts to see the data in a nicer way and also to be able to compute the calories burned based on the physical activity.

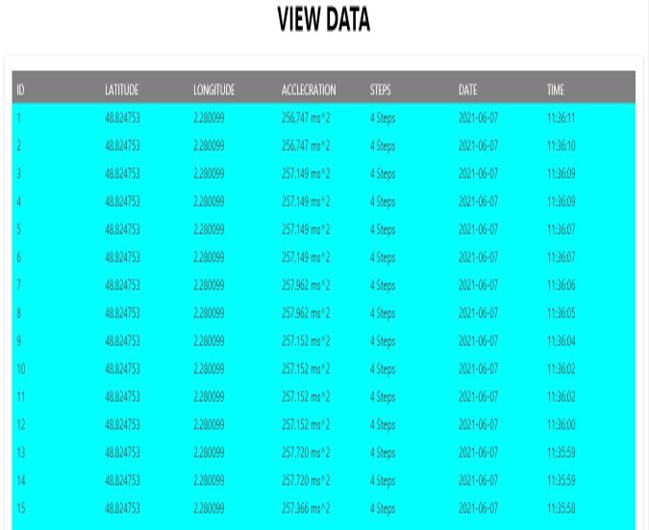
The Arduino is connected to the same local network (during to the constraints we had connection with mobile hotspot) and is given the information about SSID and password in the code. After it gets connected we will send the data from the Arduino to the database by a post method and then in the webpage we use a get method to display the values into a table. The values are displayed the most recently first and after that the rest. We have sampled 8000 values into our database from the Arduino and the communication it is working despite the fact that we were using mobile hotspot connectivity to have access to the ports.

****

**Outputs from the Arduino to the database**

****

As we can see we have stored successfully the data from the Arduino to the WIFI client’s database.



* **Pedometer**

We can demonstrate this product like wrist band, Handy unit or Rolling belt unit. It has sensor in it which works by battery.

1. **Patients shall use this product and Output from this sensor make sure their safety.**

* Step counting in wearable device based on BMA456
* Allows low-noise measurement of accelerations in three perpendicular axes
* Featuring 16-bit digital resolution and embedded intelligence

**Electrical/Interface Requirements:**

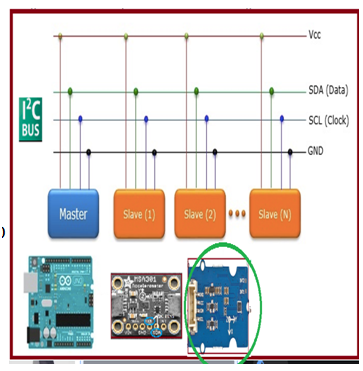
1. Operating voltage 3.3V/5V
2. Supports I2C bus protocol
3. Weight 3.2g
4. Options in sensitivity
5. Fancy (Display, colored product)



**I2C Interface Details of Pedometer**

Inter integrated circuits widely used for attaching lower-speed peripheral chips to microcontrollers. I2C uses only two bidirectional communication SDA and SCL.

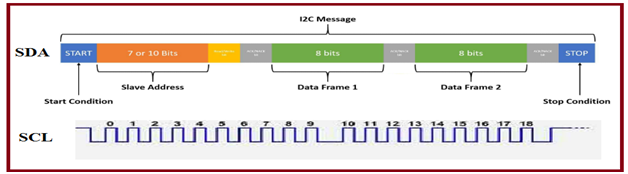
* Serial Data Line (SDA) and
* Serial Clock Line (SCL)
* Typical voltages used are +5 V or +3.3 V
* SCL and SDA as signal lines.
* Both lines are connected to VDDIO externally via pull-up resistors, so that they are pulled high when the bus is free.



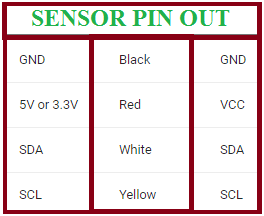
At start condition, SCL is high and SDA has a falling edge. Then the slave address is sent.

After the 7 address bits, the direction control bit R/W selects the read or write operation.

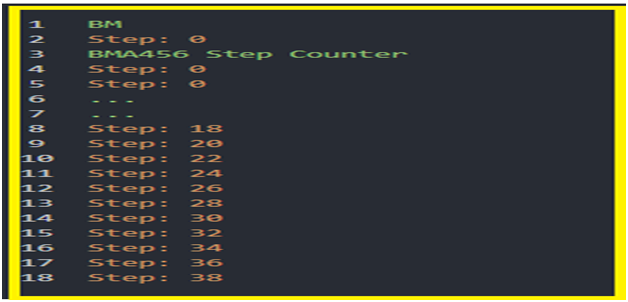
When a slave device recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle.

At stop condition, SCL is also high, but SDA has a rising edge. Data must be held stable at SDA when SCL is high. Data can change value at SDA only when SCL is low.  
  


**Integration of Pedometer**

1. Terminate SDA, SCL, Vcc and GND pins of sensor to controller as per the standard color code
2. Download and integrate the BMA456 lib will enable Communication between sensor and Arduino.
3. Define the sensor (slave) device address in the program, **0x19 device address**
4. Initialize bma456 library
5. Enable the step counter access
6. Print the steps in loop with required delay   
     
   

**Result of Pedometer**

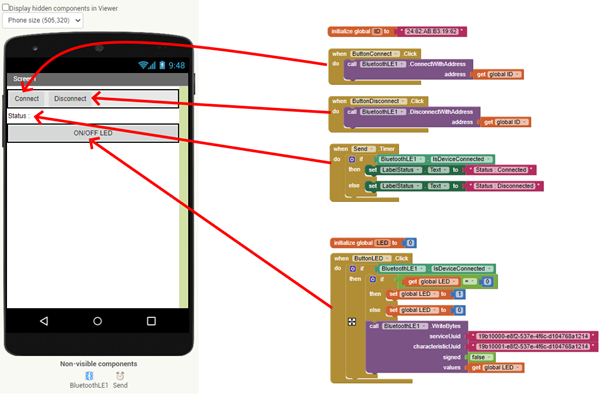
Access the step counter sensor by rotating or Movement by 45 deg, we shall able to see the output in the serial Port monitor   
  


* **Bluetooth app connection**

The objective of this feature was to transmit the information from the sensors to an android application. So, the first goal was to have the fall detection information on the smartphone.

For this we need to use Bluetooth Low Energy (BLE) technology which has a longer range than normal Bluetooth and less latency.

To create this application, we used MIT app inventor because this application editor allows you to develop android applications without the need to code. It works with a block design system, so we assign functions to objects drawn using a block system.

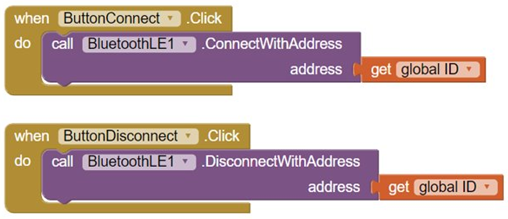


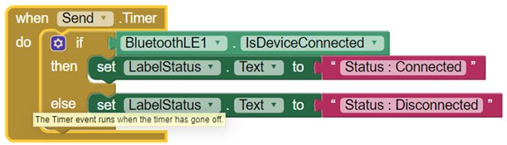
For the example, we try to turn on and off a LED remotely.

So first we need to connect to the MAC address of the Arduino board.

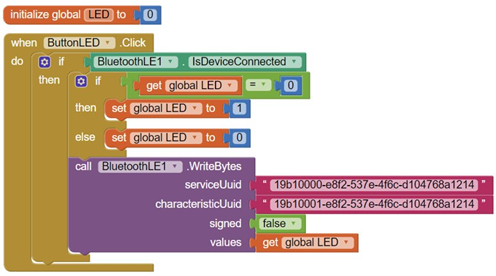


Then we create a bluetooth service to display the connection status.





After that we create a UUID service that will allow us to link to the LED precisely. And finally we write the value 1 or 0 to turn it on or off by pressing the "ON/OFF LED" button.



**Conclusion**

To conclude on our project, it is not a perfect product, we need to improve it, for example the amplification part of the speaker and removing noise from the microphone, improving charging circuit time and separating the charging circuit, integrating pedometer from the built-in accelerometer from Arduino Uno WIFI rev 2 provided, because we had problems with the library and adding more interactive functionalities to the display with buttons, menus to be more user-friendly. Also, the Bluetooth part we have tested it with light blue application on iPhone and the connection with the Arduino allowed for testing to light an LED by writing the values in hex 0x0 to turn off or 0x1 to turn on the LED and we have created a service and a characteristic so that the application subscribes to the service as in the MQTT structure. As an improvement we were thinking also about the BLE security part, to implement the encryption part of the communication, since the board has a module specially for the security part (ATECC608A). We have learnt to work as a project, despite of the problems with the coordination from the beginning we were able to coordinate at the end and deliver a functional product.