Arda Can Ertay
CID: 00863076

Prototype Design

1) Introduction:

While the PCB for the final circuit is being designed, a prototype should be built and tested. A working prototype would allow us to verify the correctness of the PCB design and to use some machine learning techniques for the discrimination of fall and not fall.

This report is composed of three sections which are Procedure, Component Selection, and Software. In the Procedure part, the method I have used to build the prototype is explained. In the Components part, the reasons for the component selections to achieve reliable, effective, and power efficient way of gathering data are stated. Lastly, in the Software part, the code used to implement the procedure is explained.

2) Procedure:

First, yaw, pitch, roll and acceleration values in x, y and z directions should be recorded and sent to a microcontroller. Then the data should be sent to a computer, so that machine learning algorithms can easily be applied using MATLAB.

The easiest way is to use a long cable connecting the microcontroller and the computer. The advantages are that the method implementation does not take a lot of time and that the real-time analysis becomes possible. However, the drawback is that this implementation limits the motion autonomy of the circuit. Sending the data wirelessly is another approach, and it increases the motion autonomy greatly while still allowing real time processing. On the other hand, building a wireless system is more time consuming. The third alternative is storing the data in a removable memory and plugging it to the computer when the testing is finished. This would bring complete motion autonomy and ease in implementation. On the other hand, real-time processing would not be possible.

The complete motion autonomy is a priority for testing purposes, because the prototype will be tested on a cyclist’s helmet while cycling. This makes the third method much more favourable than the other two methods. However, this method would not allow us to monitor the data on the computer in real time, making it hard to map each small instance to the relevant data. The other methods can be implemented as well in the future if real time data monitoring is considered useful for the tests not requiring complete motion autonomy.

To realise the third method, a microprocessor, a sensor, a battery, and a separate memory unit is needed. Adding a switch can help to determine the start and the end of the instance to be recorded to avoid dummy data. Time interval for testing can be kept short as well so that the individual small instances can be matched with the data. If this is not enough, the whole instance can be recorded by an individual camera so as to match the small instances and data representations better.

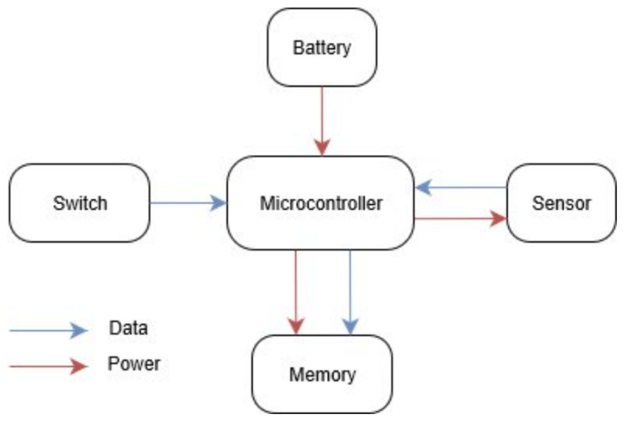


Figure 1: *Prototype Plan*

3) Components

3.1) Microcontroller

The microcontroller will be the brain of the prototype and it will get the data from the sensor and send it to the separate memory unit. Arduino platform is a good choice, because it is very easy to find the libraries to be able to read sensor data and write to the memory. Processor speed is not a priority for now, because the aim of the microcontroller is just to convey data from the sensor to the memory for now. Therefore 16 MHz Arduino [1] Uno seems to be a good choice, although a higher frequency microcontroller can be used for the final circuit if the final algorithms requires higher performance.

On the other hand Arduino Uno does have a little flash memory to store the main code and the libraries to be used. The flash memory of Arduino Uno is 32 kB and its RAM is 2kB [2]. Since lots of variables are not needed for the purpose of this project RAM is not a very big concern. However, it is further shown in the Software section that the programming space of Uno is not enough to store the main code and the libraries. Therefore, Arduino Mega having a clock frequency of 16 MHz, a flash memory of 256 kB, and a RAM of 8kB fulfils the specifications for this task.

The microcontroller in the PCB design can be changed to restrict power consumption of the overall chip, so as to keep the battery size small and increase the interval HELMO does not require charging.

3.2) Sensor

Yaw, pitch, roll and acceleration in the 3 planes should be measured to be used by the machine learning algorithms. Accelerometer can be used to measure acceleration and gyroscope can be used to measure angular velocity. Angular velocity can then be integrated to give yaw pitch and roll. Magnetometers can be used as an additional component in order to help to increase the performance of yaw pitch and roll calculation [3]. However to make the orientation values more accurate a Kalman Filter should be used together with the accelerometer and the gyroscope [4]. Hopefully, the IMU sensors which cover all of these futures are smaller and cheaper these days.

Arda Can Ertay
CID: 00863076

The IMU sensor we decided to use for our project is MPU 9250. “The MPU-9250, delivered in a 3x3x1mm QFN package, is the world’s smallest 9-axis motion tracking device.” [5] The size of the overall chip is very important for our project as the chip will be embedded in a helmet. An area of 3 mm^2 is sufficiently small and leaves out a lot of space for the other components. For the prototyping purposes, the sensor will be used on a breakout board. This sensor has a power consumption of 9.3 uA [5] which is much lower than that of the ATmega2560 used in Arduino Mega [7]. At least power consumption of MPU 9250 is not a limiting factor. MPU 9250 can give an output data frequency of up to 32 KHz which is definitely enough to detect the fall and the characteristic futures of cycling [6]. This frequency can be decreased to limit the power consumption by the microcontroller (decreased sampling frequency).

If this sensor does not give the desired performance, or if there turns out to be a better IMU sensor for this purpose, the sensor used in the PCB design can be changed easily and this is one of the aims of prototyping

3.3) Memory

The last product will not use external memory, because the plan is to save the data in an array for a small time window to determine fall or not fall to be sent to the mobile application. It should be noted that RAM comes into play this time.

However, for the prototyping purposes, size of Ram is not important, because an external memory will be used. Since this method will not be used for the last product, we don’t have to care about the power constraints of the memory units. A cheap SD card adapter for Arduino can be used for this purpose. After testing, the SD card can be removed from the Arduino adapter and connected to a USB adaptor. In this way the data can be carried to the MATLAB environment.

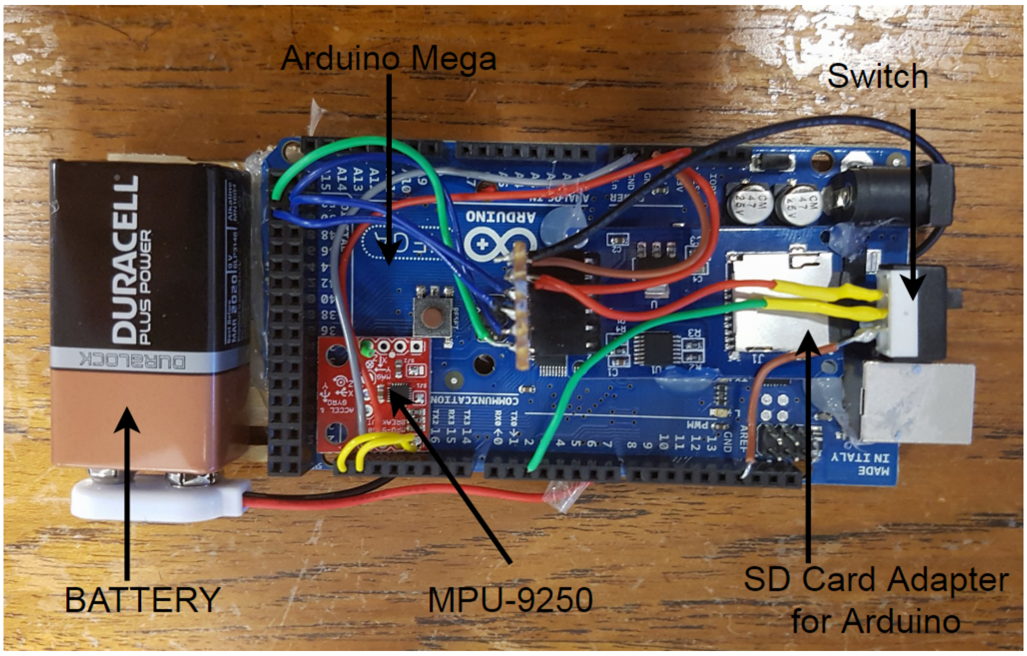


Figure 2: *Prototype*

Arda Can Ertay
CID: 00863076

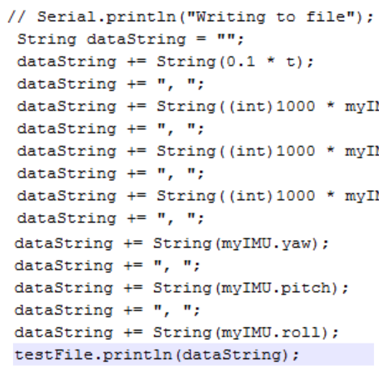
3.4) Battery

Battery to be used for the last design is not examined yet. The battery is chosen so that the prototype can be tested reliably. The Arduino Mega Board requires an external voltage between 6-12 V. The battery should also be small so that it doesn't take up a lot of space for the prototyping purposes. Therefore a 9V MN1604 Duracell battery is good enough for now.

4) Software

I used some libraries to read sensor values from the IMU sensor and to write the data to the SD card. I used quaternionFilters.h and MPU9250.h libraries to read form the sensor. I used SPI.h and SD.h standard libraries to upload the data to the SD card. I also found a template sketch to help me read the sensor values.

The only thing I did was making strings out of yaw pitch raw and 3D acceleration data. I also added a physical interrupt controlled by a switch to change a flag in order for the microcontroller to start reading for a specific amount of time to prevent dummy data. Total programming space used was 35 kB which was more than Arduino Uno flash memory, but compatible with the Arduino Mega flash memory.



```
// Serial.println("Writing to file");
 String dataString = "";
 dataString += String(0.1 * t);
 dataString += ", ";
 dataString += String((int)1000 * myIMU.ax);
 dataString += ", ";
 dataString += String((int)1000 * myIMU.ay);
 dataString += ", ";
 dataString += String((int)1000 * myIMU.az);
 dataString += ", ";
 dataString += String(myIMU.yaw);
 dataString += ", ";
 dataString += String(myIMU.pitch);
 dataString += ", ";
 dataString += String(myIMU.roll);
 testFile.println(dataString);
```

Figure 3: *Code for converting the data to string and writing it to SD card*

4) References

1) Arduino UNO R3 (2017) Available at: http://www.hobbytronics.co.uk/arduino-uno-r3 (Accessed: 24 February 2017).

2) Arduino (2017) Available at: https://www.arduino.cc/en/Tutorial/Memory (Accessed: 24 February 2017).

3) Inertial measurement unit (2016) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Inertial_measurement_unit (Accessed: 24 February 2017).

4) Luinge, H.J. and Veltink, P.H. (2005) 'Measuring orientation of human body segments using miniature gyroscopes and accelerometers', Medical & Biological Engineering & Computing, 43(2), pp. 273–282. doi: 10.1007/bf02345966.

5) InvenSense (2017) InvenSense. Available at: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/ (Accessed: 24 February 2017).

6) Alex (2016) Inertial sensor comparison MPU6000 vs MPU6050 vs MPU6500 vs ICM20602. Available at: http://blog.dronetrest.com/inertial-sensor-comparison-mpu6000-vs-mpu6050-vs-mpu6500-vs-icm20602/ (Accessed: 24 February 2017).

7) (No Date) Available at: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf (Accessed: 24 February 2017).