LUT
Department of Electrical Engineering
BL40A2010 Introduction to IoT-Based Systems

REPORT
27.10.2018

IoT-based speed measurement with magnetometer

Jouni Kortelainen 0418271
Janne Kauppi 0413506
Lauri Seppänen 0438248

TABLE OF CONTENTS

1. INTRODUCTION

In this project an IoT based velocity measurement was designed and implemented. The goal was to make a wireless sensor that an end-user could control over the internet. The system was tested with a stationary bicycle and tuned to measure a range of 6 … 60 km/h.
The system consists of a sensor, server, database and a frontend. As the sensor, LSM9DS1 was used to measure the density of magnetic flux. Raspberry PI model 3 B+ was used as the server and MongoDB, hosted at mLab, was used as the database. Frontend was run on a PC. The server was responsible for reading the measurement data, processing it and sending it to the frontend and saving it to the database. Database was used for history gathering. Frontend was used only for displaying the live data and history. The basic block diagram of the system is shown in Figure 1.

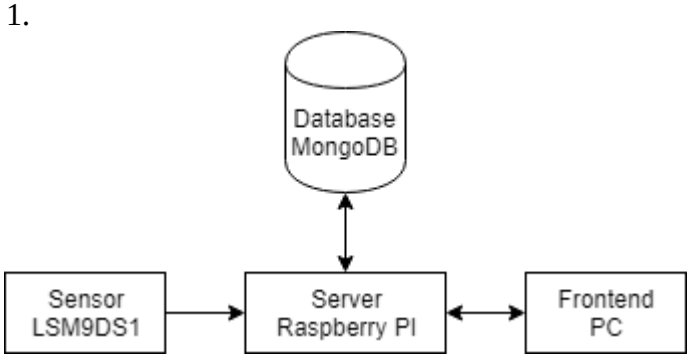

Figure 1. Block diagram of the system.

The work division was as follows:

- Jouni Kortelainen: server, measurement, testing
- Janne Kauppi: driver, database
- Lauri Seppänen: frontend

2. MEASURING VELOCITY

Measuring velocity is hard, because it cannot be measured directly, but it must be calculated from other measurements. In this project, the rotation of a wheel was monitored with a magnetometer, which detected each revolution as a magnet, secured onto a spoke, passes the sensor.

2.1 Profile of magnetic flux density

The general profile of the magnetometer's measurement can be seen from Figure 2. On low speed, we see clear spikes in the measurement when the magnet is near the sensor and on high speed some of the spikes are inverted and the magnitude is generally lower. Both profiles have a clear baseline.

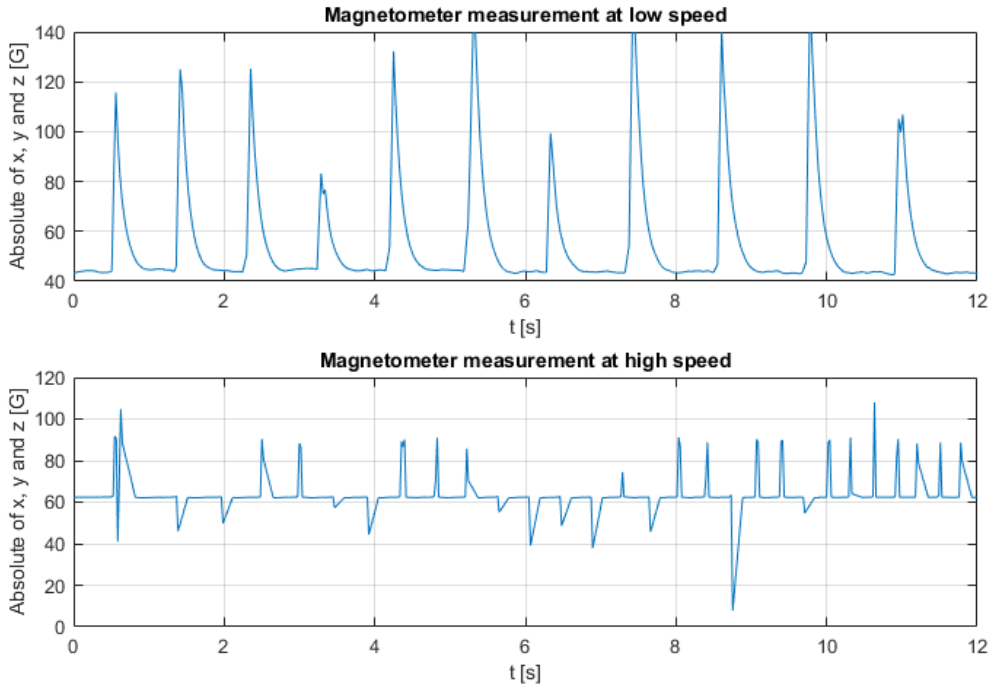

Figure 2. Profile of the absolute value of the magnetometer measurement across the x-, y- and z- axis. We can see clear spikes in the measurement as the magnet moves past the sensor.

2.2 Peak detection

The peak detection is done by finding the baseline and dividing it into two areas, peak zone and baseline zone. As the measurement goes outside of the baseline zone, a peak is detected, and the next peak can be detected only after a measurement has landed in the baseline zone. In Figure 3, a threshold value of 0.02 divides the measurement range into peak zones and a baseline zone. Peak zone for the upright peaks is $> 63.32 \cdot 1.02\ G$ and for the inverted peaks is $< 63.32 \cdot 0.98\ G$. The baseline zone is between the two peak zones.

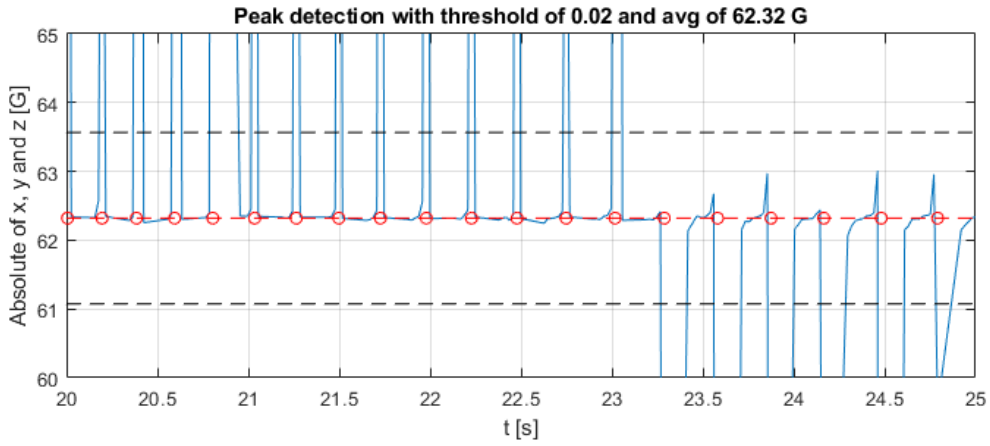

Figure 3. Peak detection. Each red circle represents a detected peak.

2.3 Calculating velocity

The velocity can be calculated by

$$v = \frac{s}{\Delta T} = \frac{\pi d}{\Delta T}, \tag{1}$$

where d is the diameter of the wheel and ΔT is time difference between detected peaks. As the velocity increases, ΔT and the resolution of the velocity measurement decreases rapidly. For this project, a ΔT range of 1500ms to 130ms was chosen, meaning the velocity range was 5.3 km/h to 60.9km/h. The ΔT as a function of velocity is depicted in Figure 4 when the diameter was 0.7.

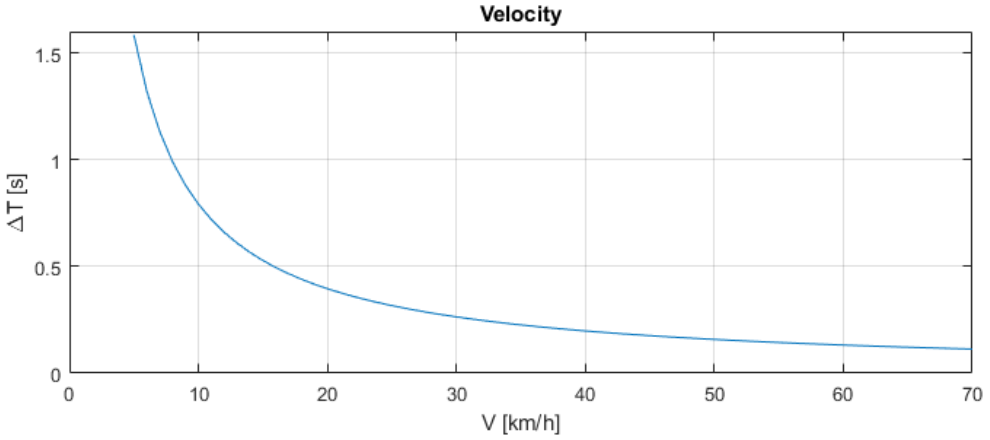

Figure 4. ΔT as a function of velocity. High velocities are hard to measure accurately since the change in ΔT decreases.

3. IMPLEMENTATION

3.1 Sensor

LSM9DS1 contains 3D accelerometer, gyroscope and magnetometer. For this project only the magnetometer is used. Communication between the sensor and the Raspberry PI can be done using I^2C bus.

During the initialization of the sensor, performance mode for x and y axis is set to ultra-high, the output data rate is set to 80 Hz and the sensor is set to continuous-conversion mode.

The measurement data is divided into six registers, two for each of the axis and the data in the registers is stored as two's complement. When new data is available, it is read and the combined absolute value of x, y and z is calculated.

3.2 Server

The server follows the MEAN-stack design pattern where Node.js is used as the runtime environment, Express as the web application framework and MongoDB as the database. Angular is replaced with a desktop-frontend made with PyQT.

The server has two controllers, sensor and data, which control the API request sent to the server. The api/sensor path can be used to turn the sensor on and off and api/sensor/latest returns the latest value from the velocity calculation module. Api/data/t1/t2 is used for history search, where the user can specify start and end of the history query with timestamps t1 and t2.

History gathering is done with a MongoDb database, hosted at mLab. The documents contain id, value and a timestamp.

3.3 Velocity measurement

The general flow of the measurement is described in with an asynchronous state machine in Figure 5. If too many samples are measured, and no peaks are detected, the module resets, records a zero velocity and clears the saved timestamps. The number of allowed samples changes according to the last measured velocity, to keep the detection of missed peaks working in low and high speeds.

The validity of timestamps is tested by calculating the ΔT. If the ΔT is within the allowed range, the velocity is calculated and saved and if not, the module resets.

Occasional measurement errors are filtered out with a median filter with a window size of 5. Small window size keeps the lag low while removing most of the errors.

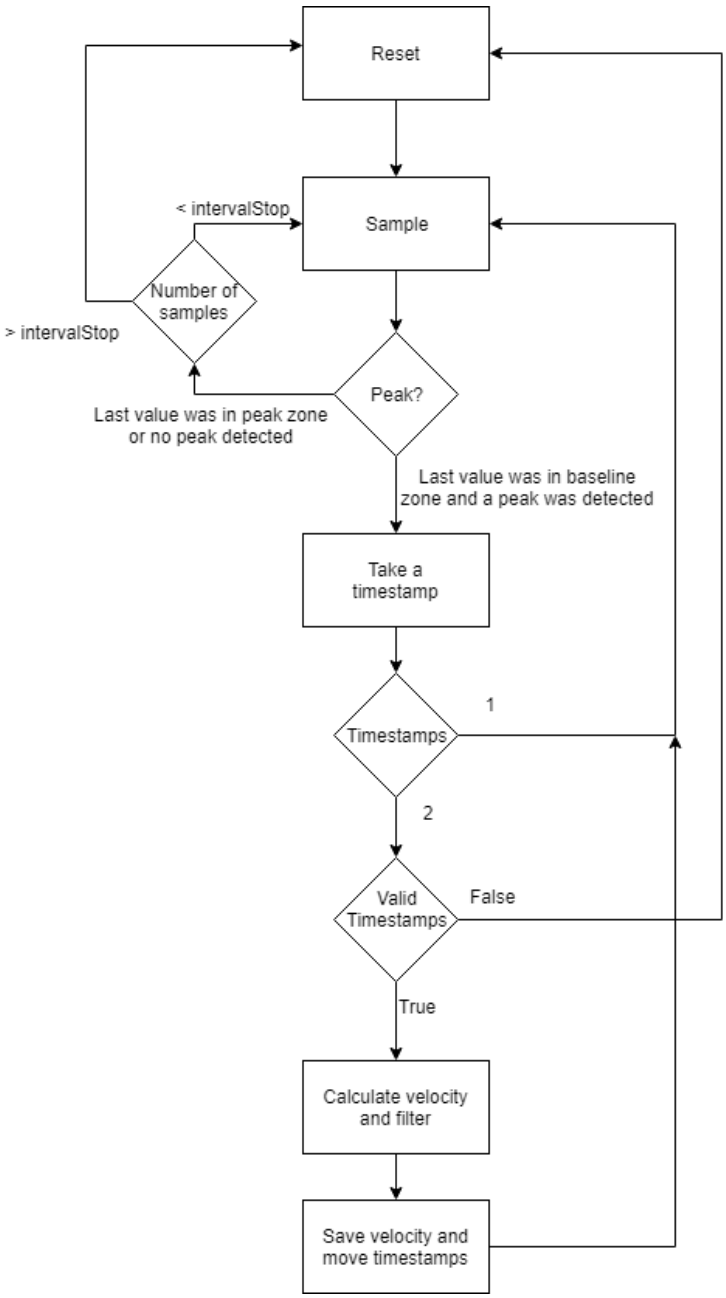

Figure 5. Asynchronous State Machine of the measurement process.

3.4 Frontend

Frontend was made with Python. The graphical user interface was designed using Qt Designer and then translated to Python and used with PyQt5. The frontend requests data from backend and can plot either live data or request history within a specified date interval. Data acquisition was done in separate thread so that the interface wouldn't be frozen during live plotting. Plotting was done with PyPlot and data storing with Pandas, because with them it is easy to adjust to different timescales in x-axis. The graphical user interface can be seen in Figure 6.

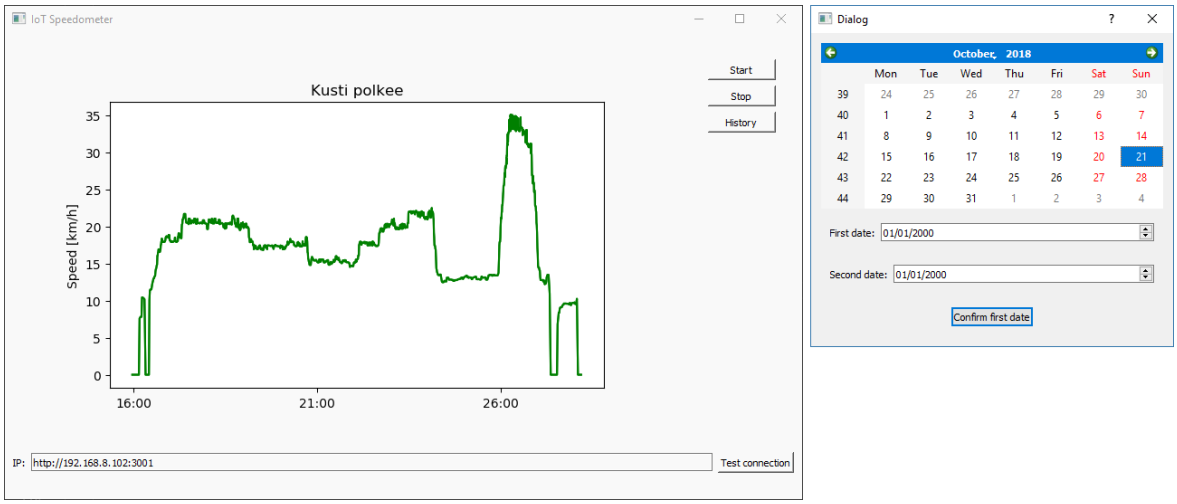

Figure 6. Frontend with a history plot and the history selection calendar.

4. TESTING AND RESULTS

The system was tested with a bicycle secured on an indoor trainer so that the back wheel, and the magnet secured on it, can rotate in place. Magnetometer and the Raspberry PI were placed next to the wheel. Raspberry PI and the frontend PC were connected to the same network.

The performance of the API was tested using a middleware called Morgan which calculates the amount of time it takes to answer to a request. Initially the datapoints were fetched from the database at the time of request. Testing different methods proved that getting the latest datapoint from the timer module was significantly faster, decreasing the average response time from around 150 ms to 1ms.

High speeds often resulted in absurd results, because the trailing measurements after a peak had oscillation. The theoretical minimum of $\Delta T$is 25 ms (317 km/h with d = 0.7) since the sampling rate of the sensor is 80 Hz and the used peak detection requires one measurement in baseline zone and two in peak zone to calculate the ΔT. By introducing a small delay of

100 ms to the peak detection, the minimum of ΔT becomes 125 ms (63 km/h with d = 0.7) and the errors decrease while we stay in the target velocity measurement range.

After applying the threshold and the median filter, the output velocity does not contain major errors.

The result of the range test is depicted in Figure 7. From the test we can see that the measurement works all the way up to 60 km/h but velocities higher than that result in inaccurate measurements, as seen between 30 and 45 seconds. The slope from 60 km/h to 7 km/h is smooth and free of errors. The last measured velocity was 6.8 km/h at 100 seconds and after that, the wheel halted.

The range test also shows how the resolution changes over the range. The step size at 45 seconds is 4 km/h and at 90 seconds it is 0.5 km/h.

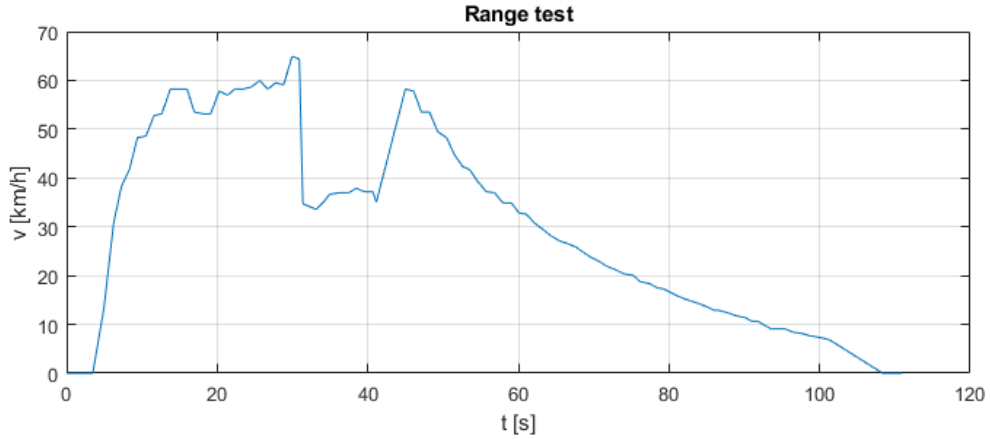

Figure 7. Range of velocity measurement. Measurements above 60 km/h resulted in errors and under 6.8 km/h resulted in zero velocity. The slope between the two extremities is error free.

5. CONCLUSION

A system for measuring speed of a bicycle was made. The system consists of a magnetometer sensor, server for handling the data, separate database for storing the data and a frontend for a user interface. Raspberry pi was used for the server, MongoDB for the database and desktop app developed with Python for the frontend. The system measures speed reliably from 7 km/h to 60 km/h, which is a good range for the application.