AGH University of Science and Technology

Metodyki projektowania i modelowania systemów 1

Speedometer with Display Data on Phone

Authors: Hubert Jabłoński
Mentor: Prof. dr hab. inż. Bogusław Cyganek

Monday 2nd September, 2024

Contents

1 Introduction

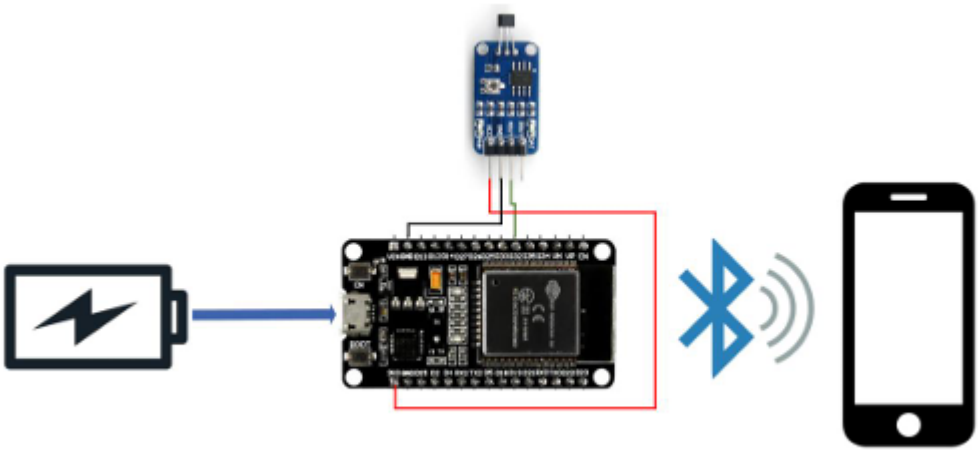

Figure 1: System connection diagram

According to the above diagram, a Hall sensor was connected to the microcontroller. The analog output of the sensor was connected to GPIO32, which is equipped with an ADC converter. The circuit was created, and a picture of it is shown below.

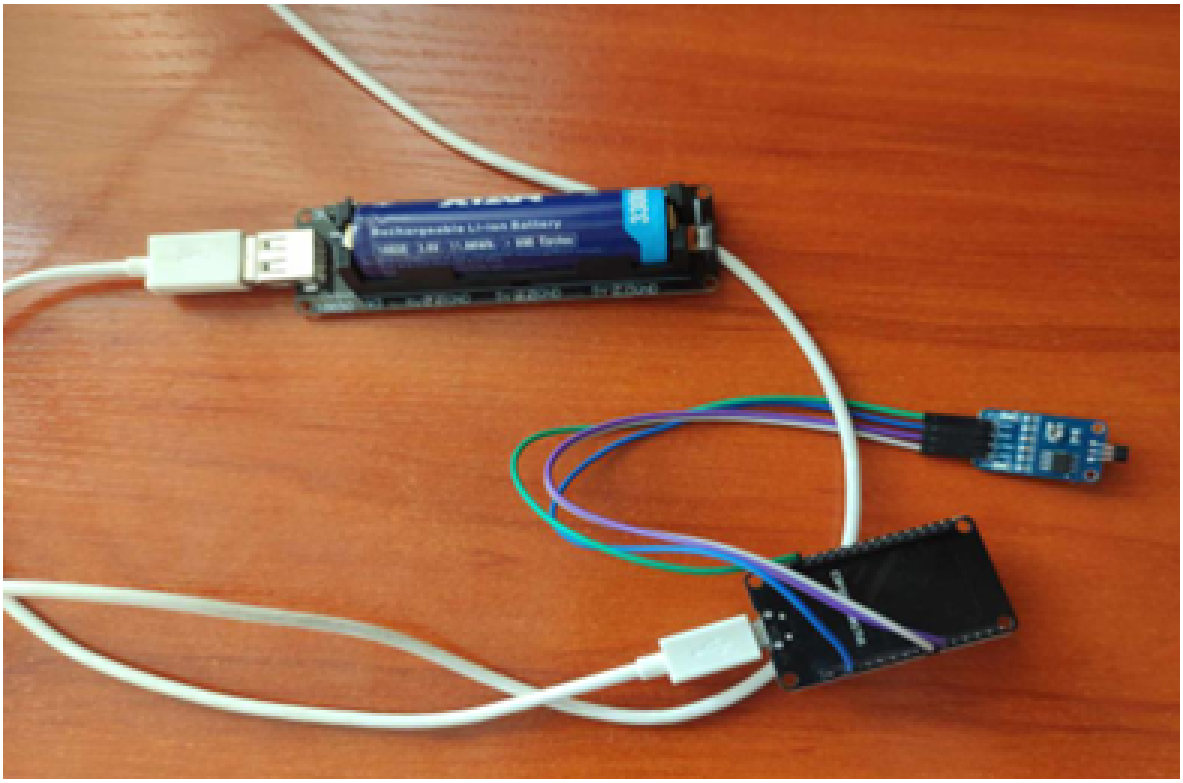

Figure 2: Photo of the completed system

The device was successfully programmed and tested by moving a magnet towards the sensor at varying speeds. The data read from the sensor was displayed both in the terminal on the computer, and later on, the sending and receiving of data on a mobile phone were tested through a mobile application. The program was used to calculate both the rotational speed and the linear speed. The device allowed for calibration, which was necessary because the application was intended to be compatible with wheels of all sizes.

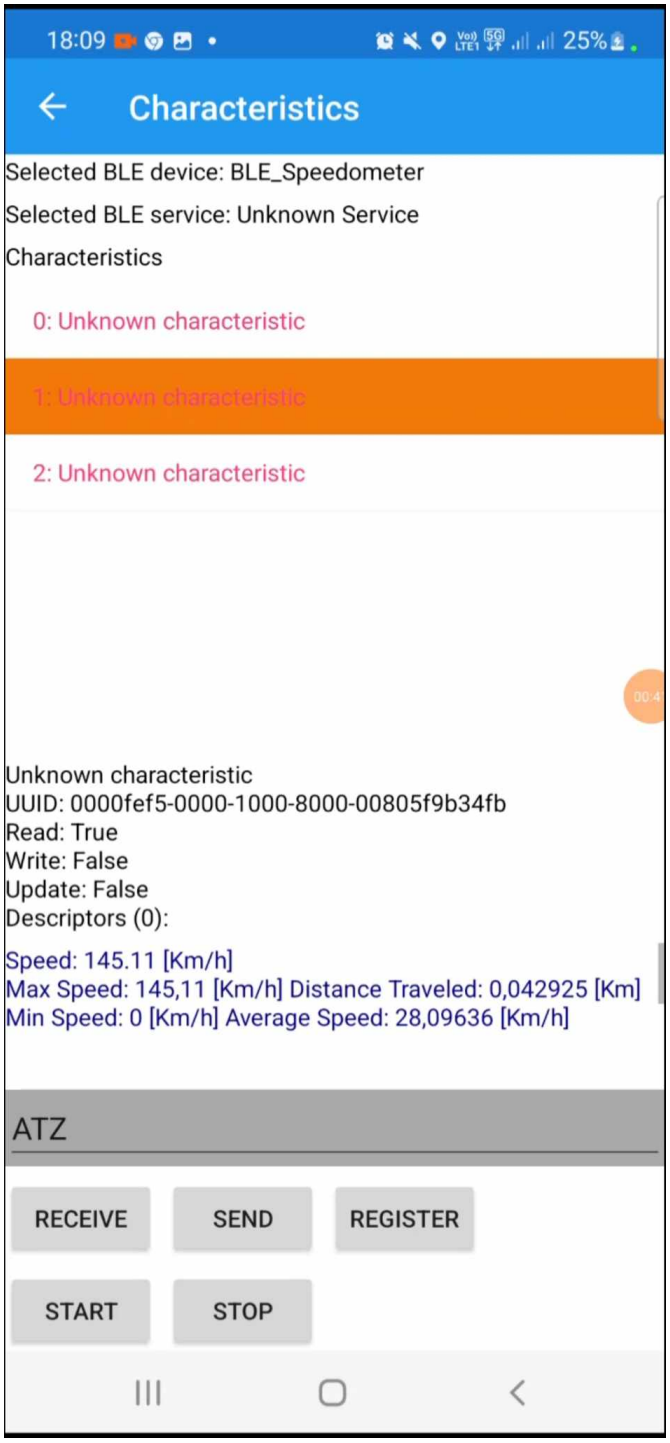

Figure 3: Android App

The results are displayed in an application created in Visual Studio using Xamarin in C. The app shows the maximum speed (maxSpeed), which is updated whenever the current speed exceeds the previously recorded value. The minimum speed (minSpeed) is updated when the current speed is lower than the previously recorded minimum. The average speed (averageSpeed) is calculated as the ratio of the total sum of all recorded speeds (totalSpeed) to the number of measurements (speedCount). The app also calculates the distance traveled.

2 Components

- **Hall Sensor:** AH49E on Waveshare module
- **Microcontroller:** ESP32
- **Power Module:** Powerbank module with 18650mAh battery holder
- **Phone:** Android smartphone

3 Esp Code

The microcontroller was programmed using ESP-IDF with FreeRTOS for task management. The main function in `Speedometer.c` handles Bluetooth initialization and sensor data processing. BLE functionality is implemented in `nimBLE.c`, with tasks created to manage BLE operations and initialization via the BOOT button.

The `adc_sensor.c` file manages continuous ADC readings and time measurements between sensor signals. It detects when the Hall sensor's signal exceeds a 1900mV threshold, calculates RPM, and converts it to linear speed (km/h) based on the wheel diameter. The program uses FreeRTOS for task synchronization and interrupts for ADC conversion completion.

The general flow of the application, which aids in understanding its functionality, is shown below:

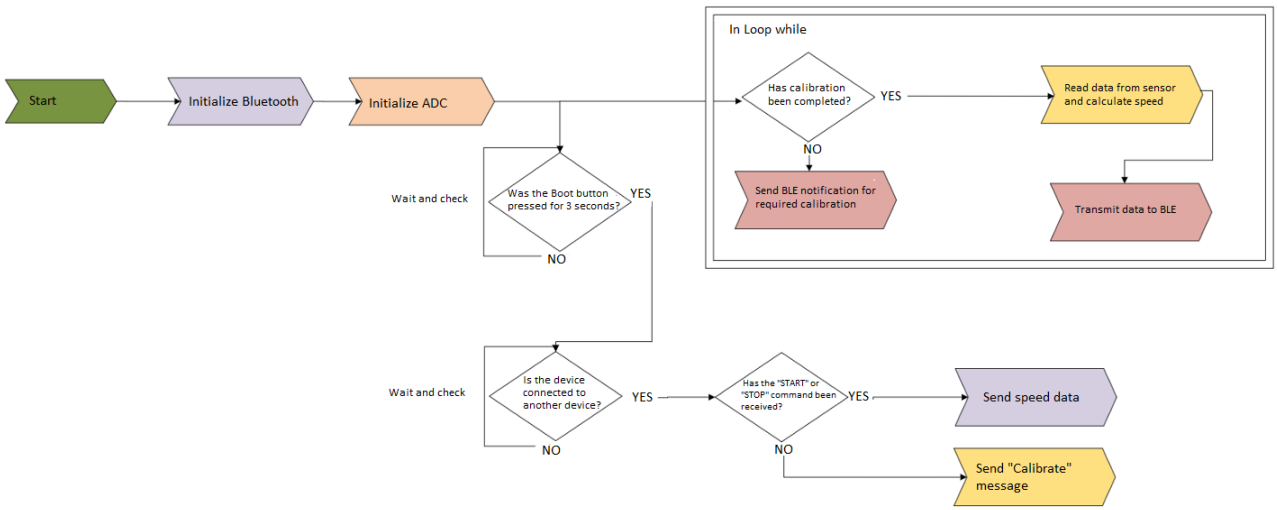

Figure 4: Flowchart of how the code works on esp

4 Android App Code

The provided code is for a mobile Bluetooth Low Energy (BLE) client built with Xamarin.Forms. The app is designed to connect with BLE devices, read and write their characteristics, and monitor data in real-time. My focus in this project was on developing an Android app that displays data sent from the ESP32 device. This app not only visualizes the current speed but also provides additional information critical for accurate monitoring and analysis of activity.

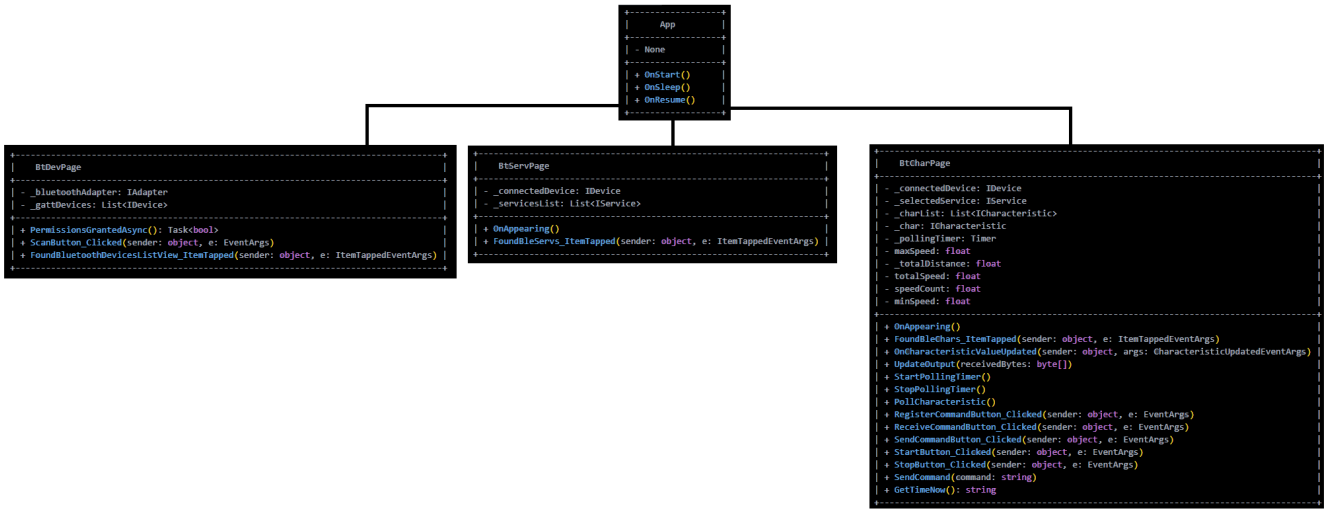

Figure 5: Android App Diagram

During the project, the following sources were used:

- https://docs.docker.com/get-started/get-docker/ [Access 02.09.2024]
- https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads [Access 02.09.2024]
- https://github.com/espressif/esptool/releases [Access 02.09.2024]
- Espressif Systems. "API Reference - NimBLE-based Host APIs." [Access 02.09.2024]
- Espressif Systems. " API Reference - Analog to Digital Converter (ADC) Continuous Mode Driver" [Access 02.09.2024]
- BCD. "AH49E Datasheet" [Access 02.09.2024]
- Waveshare. "Hall Sensor" [Access 02.09.2024]
- Espressif. "ESP32 Datasheet" [Access 02.09.2024]
- Forbot. "Co warto wiedzieć?" [Access 02.09.2024]