# Wireless Gesture Controlled Applications

Aman Kumar Rawat, Antonia Sakellariou,
Nidhi Chakrabhavi Basavaraju


Course: Programming Embedded Systems Project

Date : Monday 30th May, 2022

# Contents

**1 System Description** **2**
- 1.1 Overview . . . 2
- 1.2 Usage scenarios . . . 2
- 1.3 Applications . . . 3
- 1.4 Structure . . . 3

**2 Design and Implementation** **4**
- 2.1 Hardware . . . 4
- 2.2 Software components/structure . . . 5
- 2.3 Theory and algorithms . . . 6
- 2.4 Implementation details . . . 6

**3 Experiments** **8**
- 3.1 Test scenarios . . . 8
- 3.2 Results . . . 10

**4 Outcomes** **11**
- 4.1 Challenges . . . 11
- 4.2 Limitations . . . 11
- 4.3 Insights . . . 11
- 4.4 Future work . . . 12

# 1 System Description

## 1.1 Overview

The system described in this report contains a gesture controlled menu, which allows the user to select between two options by moving a motion sensor up and down. The menu consists of two options:

1. Recognition of shapes created by hand movements of the user.
   The user draws shapes in the air by moving the sensor, and the system recognizes the shape and prints it on the screen.

2. Simulation of a gesture controlled car.
   The user moves the sensor up down left or right, and the car on the screen moves according to these movements.

The project can be found in `https://github.com/AmanKrRawat/PES_Gesture_control`.

## 1.2 Usage scenarios

**Scenario 1**
The user by moving the motion sensor down and pressing the selection button selects the **CAR GAME** option. Then moves the motion sensor up/down/right/left, and the car on the screen moves in the same direction. The user, can either continue moving the motion sensor in any direction, so that the car keeps moving, or press the back button and return to the menu screen.

**Scenario 2**
The user by moving the motion sensor up and pressing the selection button selects the **SHAPE** option. Then moves the motion sensor drawing a square/line/checkmark on the air, and the shape is printed on the screen. The user, can either draw another shape on the air, which will then be printed on the screen, or press the back button and return to the menu screen.

## 1.3 Applications

Some possible applications of this system in real life can be:

1. Sound volume control
   The user by moving the motion sensor up or down can control the sound level of an electronic device.

2. Controlling drones
   The user, by moving the motion sensor up down left or right, moves a drone in the same direction.

3. Controlling toys
   Same as controlling drones. Example: Toy car.

4. Robotic hand movements
   The user, by moving the motion sensor up down left or right, moves the robotic hand in the same direction.

## 1.4 Structure

The system, as shown in Fig1, consists of 2 nrf52840dk boards a motion sensor and an OLED screen.

The user can move the motion sensor, which is connected with a nrf52840dk board. The motion sensor data are sent by the nrf52840dk board via Bluetooth to the nrf52840dk board that controls the OLED screen. The output on the OLED screen depends on the movements of the motion sensor and the state of the system.


Figure 1: System overview

# 2 Design and Implementation

## 2.1 Hardware

The hardware we used for this project are the following:

1. **OLED Screen**
   - ➔ 128*32 pixels
   - ➔ I2C address for the screen is <0x3d>
   - ➔ This screen is chosen as the IC ssd1306 is supported by Zephyr and the LVGL library.
   - ➔ The screen is OLED, so only the pixels required are turned on, resulting in saving energy.

2. **MPU6050 motion sensor**
   - ➔ 6-axis inertial sensor
   - ➔ I2C address for the sensor is <0x68>
   - ➔ The sensor returns the raw values of gyroscope in "rad/sec" and of accelerometer in "g".
     - Accelerometer range: ±2, ±4, ±8, ±16 g energy
     - Gyroscope range: ±250, ±500, ±1000, ±2000 °/s

3. **Two nrf52840dk boards**
   - ➔ 64 MHz Arm Cortex-M4
   - ➔ RAM : 256 KB
   - ➔ ROM : 1 MB
   - ➔ On board Bluetooth communication
   - ➔ Buttons and LEDs
   - ➔ UART interface through virtual COM port

4. **Breadboard**

5. **Jumper wires**

## 2.2 Software components/structure

For this project, we took references from the Zephyr nrfSDK. [2]

**Software used while implementing this system:**

1. Zephyr : 3.0
2. LVGL : 8.2.0
3. Python : 3.10.0
4. Compiler : 10.3.0
5. Cmake : 3.20.0
6. Git : 2.35.1.windows.2

**Structure:**


(a) MPU6050 Software Structure (b) OLED Software Structure

Figure 2: Project repository structure

**CMakeLists.txt:**
This file describes the build options for the project. DTC_OVERLAY_FILE path and the file name need to be specified here. The files to be compiled need to be added here. The SHIELD option defines the OLED specification.

**prj.conf:**
In this file, we can turn on/off the specific settings for the project. This creates the pre-processor defines for selective compilation.

**src:**
This directory contains the source files for the project.

**nrf52840dk_nrf52840.overlay:**
This file describes the external component specifications. We can specify the following here:

- ➔ I2C baud rate.
- ➔ The pin connection for SDA and SCL.
- ➔ The I2C register base address.
- ➔ The component names

## 2.3 Theory and algorithms

**Algorithm:** Random Forest Classifier

The classifier uses multiple decision trees to compute the output. The output is the class, voted the most by the trees.[1]

## 2.4 Implementation details

**Screen Working Model:**
Fig3 describes how the screen side works.

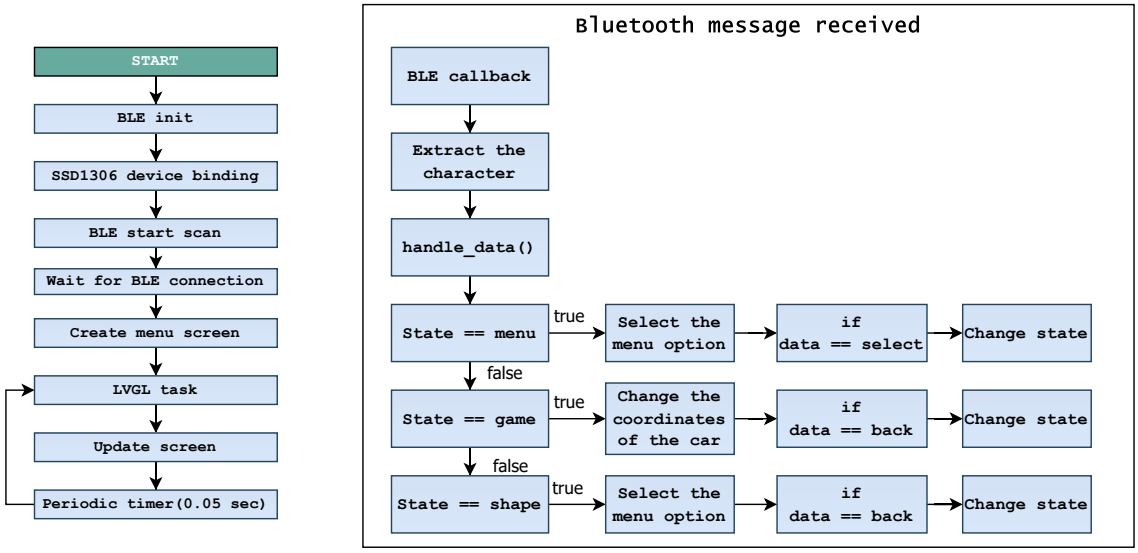


Figure 3: Screen implementation overview

**Sensor Working Model:**

Fig4 describes how the sensor side works.

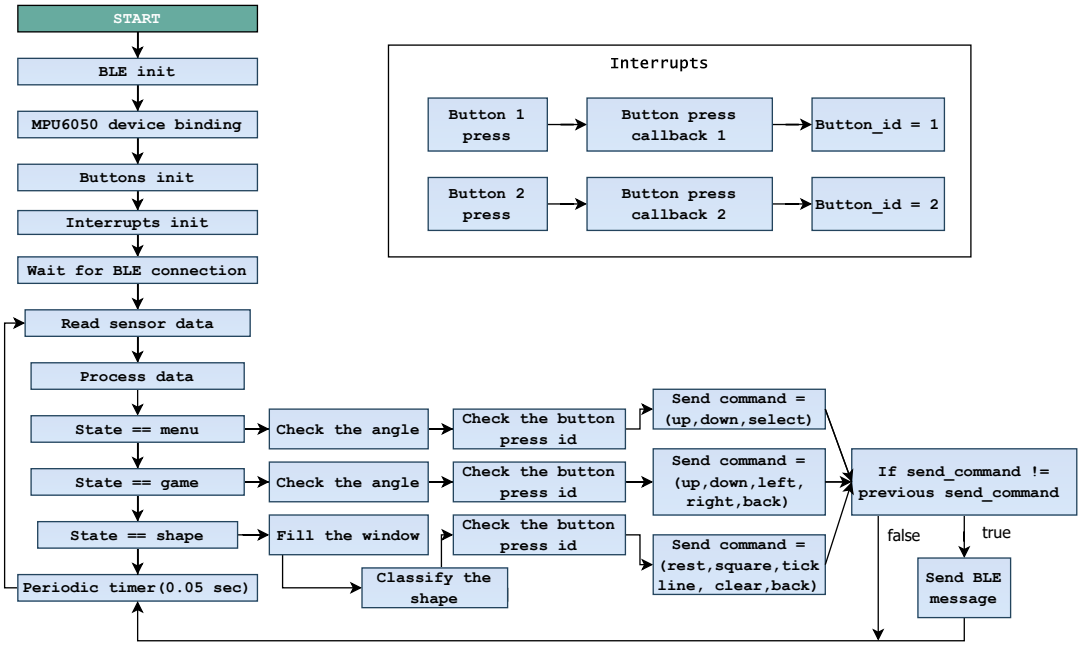


Figure 4: Sensor implementation overview

# 3 Experiments

## 3.1 Test scenarios

To test the system, we decided to check all the possible scenarios and to observe if the results we got were correct. To be exact, we tested how all the options available for each state performed.

For the **MENU** state, we tested what happened when:

- ➔ The user moved the motion sensor up. The expected outcome is that the menu selector moves up.
- ➔ The user moved the motion sensor down. The expected outcome is that the menu selector moves down.
- ➔ The user presses the select button. The expected outcome is that the option pointed by the menu selector is selected.

For the **CAR GAME** state, we tested what happened when:

- ➔ The user moved the motion sensor up. The expected outcome is that the car moves up.
- ➔ The user moved the motion sensor down. The expected outcome is that the car moves down.
- ➔ The user moved the motion sensor right. The expected outcome is that the car moves right.
- ➔ The user moved the motion sensor left. The expected outcome is that the car moves left.
- ➔ The user presses the back button. The expected outcome is that the system goes back to MENU state.

For the **SHAPE** state, we tested what happened when:

➔ The user did not move the motion sensor. The expected outcome is that nothing is drawn on the screen.

➔ The user drew a square with the motion sensor. The expected outcome is a square is drawn on the screen.

➔ The user drew a checkmark with the motion sensor. The expected outcome is a checkmark is drawn on the screen.

➔ The user drew a line with the motion sensor. The expected outcome is a line is drawn on the screen.

➔ The user presses the back button. The expected outcome is that the system goes back to MENU state.

**Example**

In Fig5, we can observe the expected behavior of the system when it is on the CAR GAME state. Specifically, in the sensor side after the movement by the user the values collected are converted into angles and compared to threshold values (defined by the ML classification algorithm). Then the message is sent by the nrf52840dk board connected to the sensor to the other board via Bluetooth. In the screen side when the message is received, the x, y values are changed for the car and the new position is printed on the screen.

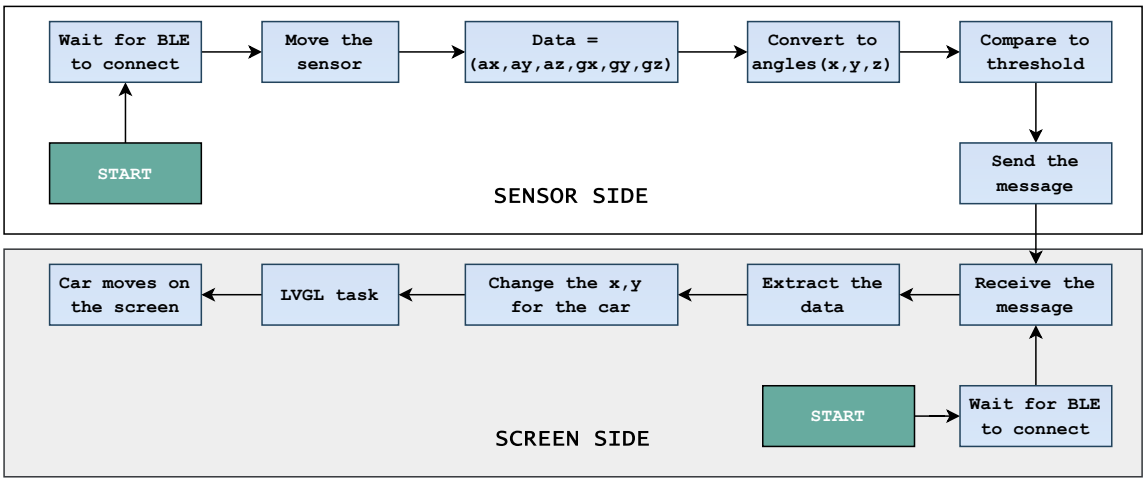


Figure 5: Car example overview

## 3.2 Results

By running a series of tests, we observed that the functionality of the system is correct. To be more specific, as shown in Table1 the system given an input always produces the expected output. The ML classifier results are shown in Fig6.

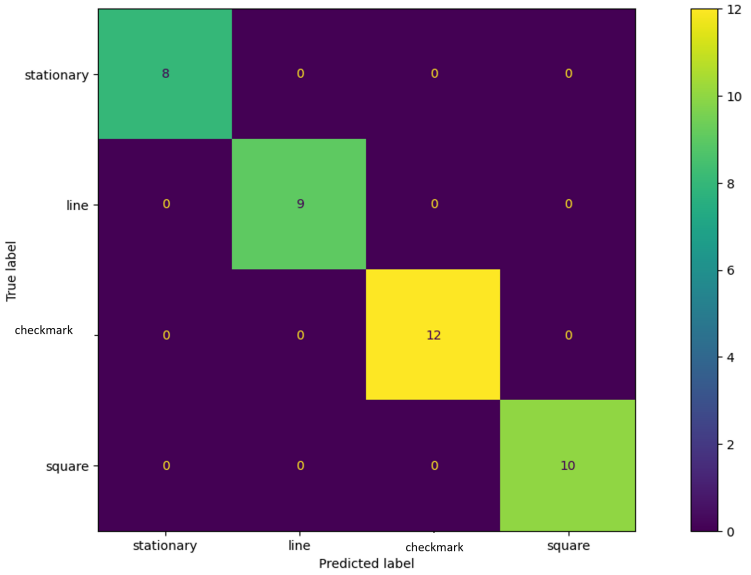


Figure 6: ML Classifier data

| State | Input | Output | Output Evaluation |
|---|---|---|---|
| Menu | Up | Move menu selector up | Correct |
| Menu | Down | Move menu selector down | Correct |
| Menu | Button press | Select menu option | Correct |
| Car game | Up | Car moves up | Correct |
| Car game | Down | Car moves down | Correct |
| Car game | Left | Car moves left | Correct |
| Car game | Right | Car moves right | Correct |
| Car game | Press back button | Go back to menu | Correct |
| Shape | Do nothing | Screen is empty | Correct |
| Shape | Draw square | Square is drawn on the screen | Correct |
| Shape | Draw checkmark | Checkmark is drawn on the screen | Correct |
| Shape | Draw line | Line is drawn on the screen | Correct |
| Shape | Press back button | Go back to menu | Correct |

Table 1: Behavior of the system

# 4 Outcomes

## 4.1 Challenges

During the implementation of this project, we faced a significant amount of challenges, which in the most part we were able to overcome.

1. To begin with, we had a problem with the readings of the motion sensor. In particular, the sensor reads in angular velocity, which was a problem for us. In order to solve this, we converted the angular velocity values to angle values. For better results, we actually calculated and used the weighted average of the accelerometer and the gyroscope converted values.

2. Secondly, there were limited resources on the usability of the LVGL library, so we had to do a lot of searching on the existing resources in order to figure out how to use it properly.

3. Lastly, we faced some difficulties with the shape recognition, which we eventually solved by using machine learning.

## 4.2 Limitations

This project has some limitations, which could have an impact on its functionality:

1. Distance is limited to 10 m as we use Bluetooth

2. Security is not guaranteed

3. Interference from other devices might be a problem

4. Low power operation is not supported

5. Bluetooth is unidirectional, which results in having to store the states in both boards

## 4.3 Insights

During this project, we learned how to work with the MPU6050 motion sensor, how to work with an OLED screen and how to use the Bluetooth of the nrf52840dk boards. We also realized how we can use machine learning algorithms to get better results and to improve our system's functionality.

## 4.4 Future work

Some possible additional features and improvements, that could be implemented in the future are:

1. Implementing the project on a touch screen, which would make it even more interactive for the user.
2. Add multiplayer mode for the car game.
3. Improve security, so that the system will not be easily impacted by other devices on the area.
4. Implement a low power solution.
5. Make the Bluetooth connection bidirectional.
6. Use the interrupt mode of the sensor.

# References

[1] Random forest classifier. `https://en.wikipedia.org/wiki/Random_forest`. Accessed: 30-05-2022.

[2] Zephyr nrfsdk. `https://github.com/nrfconnect/sdk-zephyr/tree/main/samples`. Accessed: 30-05-2022.