Drone Sim

27.06.2021

Group Members

Berke BELGİN
Ezgi KOTAN
İlkan Mert OKUL
Kamil KAYA
Yusuf AKGÜL
Baran Hasan BOZDUMAN
Muzaffer Deha PELİT
Türker TERCAN
Abdullah KÜSGÜLÜ
Zafer ALTAY
Alina KURALOVA
Furkan ÖZEV
Canberk ARICI

Consultant:

Prof.Dr. Erkan ZERGEROĞLU

Project Demonstration Link: https://youtu.be/xQJXUZrwLos
GitHub Link: https://github.com/DroneSim-Group7/DroneSim
WebSite Link: https://group7dronesim.wordpress.com/

CONTENTS

1. DRONE SIM SIMULATOR

1.1 Project Definition

1.1.1 Project Name

Name of the project - “DroneSim”

Contributors: CSE 396 GROUP 7

1.1.2 Project Overview

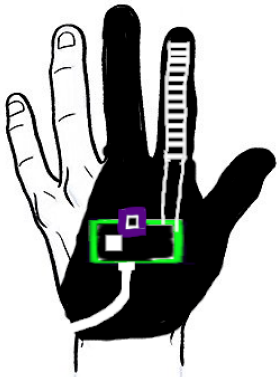
In this project, a drone is simulated with a glove or gamepad controller according to the user's will. The glove accurately tracks the pilot's hand position, finger gestures and translates these movements into commands for the drone in the simulator. These commands are direction commands to move the drone, as well as several functional commands can be added. So the drone in our simulation environment moves all the directions in the coordinate system according to commands, which comes from glove movements. Including the Glove controller hardware part, we also did the Gamepad controller part. Which means we can control and move the drone with the help of a gamepad as well. GamePad connects to the computer, we have a couple of simple buttons for controlling the drone. Drone can move all the directions in the coordinate system with the help of a gamepad as well. One of the most interesting parts in our project is the Simulation part, where we have implemented the GTU map environment for the drone. All environments including: buildings, benchs, grass, football fields, cars, roads, trees, bridges, railway and houses were reproduced by us. So we have a map of our university, where we can check the actions/movements of the drone. Drone which is controlled by a glove or gamepad, according to the user's will.

1.2 Application

1.2.1 Drone Operation

We operate and move the drone using 2 ways: using Glove Controller and GamePad Controller. For the GamePad controller we have commands which come from gamepad outputs, we accept them as inputs for the software simulation part and move the drone accordingly. In the Glove Controller part it works approximately the same way. Using a gyroscope/accelerometer sensor to sense the motion of the wrist and flexible sensors for movement of the finger we determine which direction the drone will move. We transmit these outputs to the simulator unit and use a serial communication port for communication in between.

1.2.2 Control With Glove



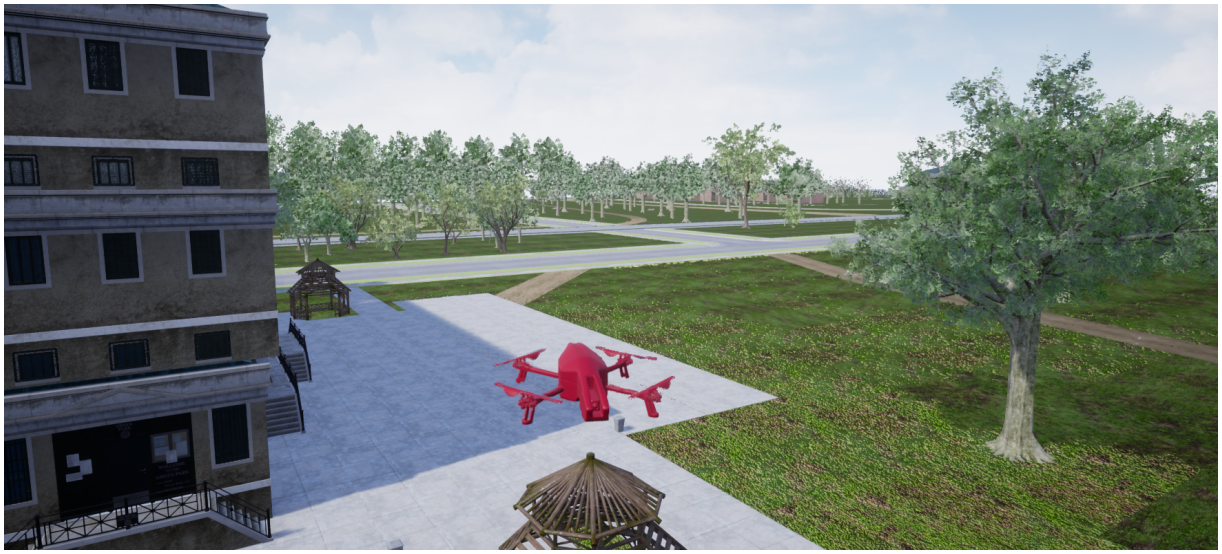
Above the picture of the first version of a glove we sketched out was attached. This glove uses a gyroscope/accelerometer for determining movements of a hand. We operate the drone using the movements of our hand. In other words, hand actions show in which direction the drone should move. For example: the user's hand movement to the right will move the drone to the right direction etc. And finger actions used as a command, transmitted by the glove to the simulator.

1.2.3 Control With GamePad

Above the picture of the approximate version of the GamePad we will use in order to operate/move the drone. Outputs from GamePad transmits to the simulator and the simulator moves in the proper direction. Left button moves the drone to the left direction etc

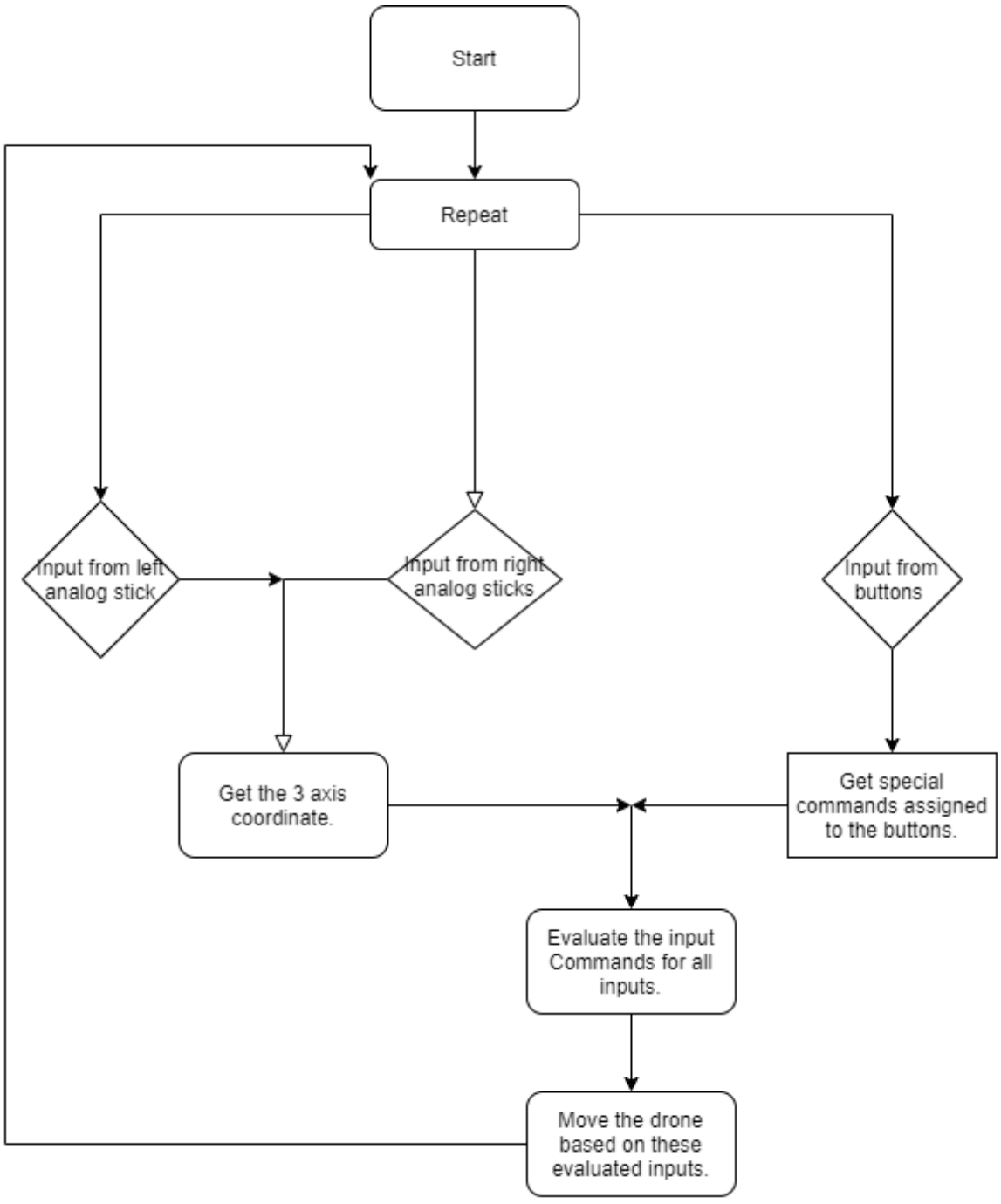
1.2.4 GTU map, Overcoming obstacles

When we operate/move the drone we meet/hit many obstacles on the way like: buildings, cars and other objects. To make everything work correctly, we have implemented everything in an appropriate way. In the GTU Map area, we were having trouble with hitting obstacles. But we were able to solve them and improve our implementation in an appropriate way. In the GTU Map, we tried to implement all buildings we have in our campus.

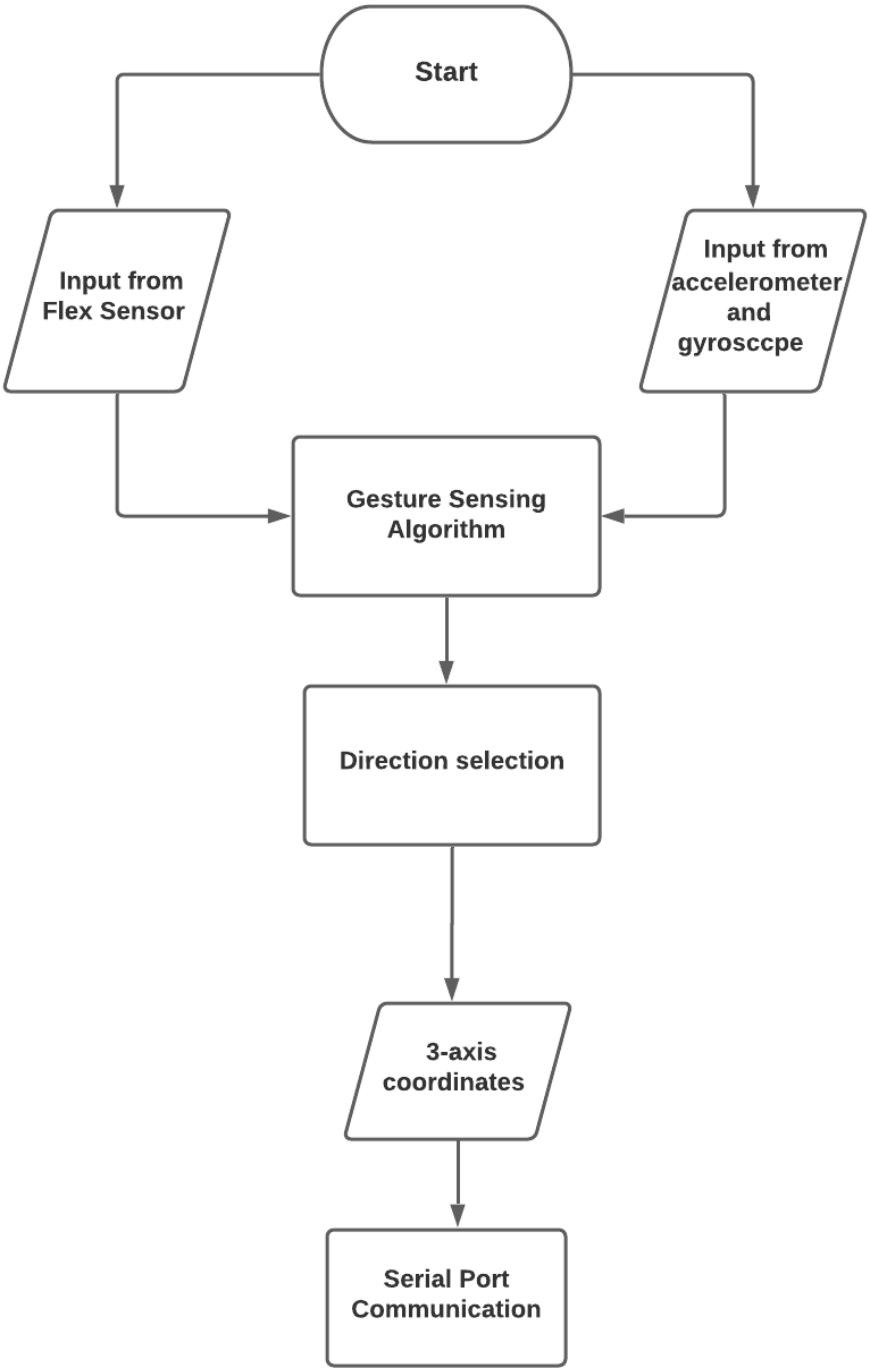


2. EXAMPLE USAGE SCENARIOS

2.1 Gamepad Flowchart



2.1 Glove Flowchart

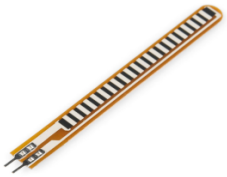


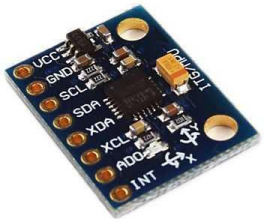
3. HARDWARE and SOFTWARE REQUIREMENTS

3.1 Hardware Requirements For Glove Controller

For the Glove Controller requirements were listed below:

- Accelerometer & Gyroscope Sensor (e.g. MPU6050)
- USB cable for connection
- Raspberry Pi Pico (RP 2040)
- LED Lights
- Flex sensor





3.2 Hardware Requirements For GamePad Controller

For the GamePad Controller requirements were listed below:

- USB cable for connection
- Raspberry Pi Pico (RP 2040)
- Thumb joysticks and buttons
- MCP-3208 8 channel 12 bit ADC converter

3.3 Common Requirements Of Glove & GamePad Controller

- USB Flash Drive containing OS and required programs
- 3.0 64GB Bootable USB Drive

3.4 Table Of Requirements Components

Name of the product	**Purpose of Usage**
Accelerometer & Gyroscope Sensor	The direction of the drone is determined by the data received from the gyroscope.
Flex Sensor	The flex sensor is used to adjust the height of the drone.
Buttons	It is used to set some commands and modes for the drone.
MCP-3208	This integrated circuit converts the analog value from the joystick to 12 bit digital values.
Thumb Joysticks	The direction of the drone is determined by the data received from the joysticks.
Raspberry Pi Pico	All these hardware work on the raspberry pi pico microprocessor and the data is processed.
USB cable for connection	It provides the power requirement for the operation of the microprocessor and the communication for transferring the data to the simulation.
Led Light	LED lights are used to indicate some status information
USB Flash Drive	A USB stick is required to boot the operating system, simulation and necessary programs.

3.5 Project Software Requirements

Software requirements for the project were listed below:

- A lightweight Windows distribution pre-installed on the flash drive.
- A simulation program that runs on Windows.

4. MODULES OF PROJECT

4.1 Definitions Of Modules

This project has five modules: Simulation Modeling , Simulation Kinematics, Gamepad Controller, Glove Controller and Communication and Integration.

4.1.1 Simulation Modelling

This module is responsible for the overall picture in the simulator. The aim of this module is to design the objects(trees, drone, buildings etc.) that will be in the simulation and the simulation world itself. For the game map our aim is to replicate the GTU campus. In the project, it is tried to create a nice simulation.

4.1.2 Simulation Kinematics

This module is responsible for the drone's behaviour in the simulation. More specifically how the drone behaves according to the given inputs, how it accelerates and decelerates, how it turns and such. The drone should have a top speed which it can’t accelerate above, should be able to hold itself in the air while no directional input is given and it should be able to interact with the world around it such as collisions and landing on stuff etc.

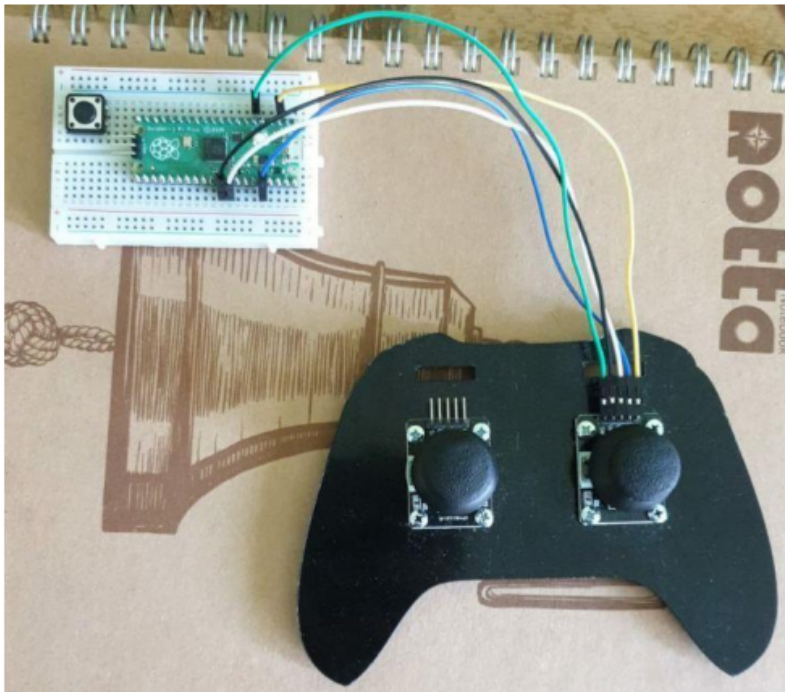
4.1.3 GamePad Controller

In the project, it was aimed to control the simulation with a PS3-like gamepad controller. Two analog sticks and some buttons on it are used to assign some functional commands.

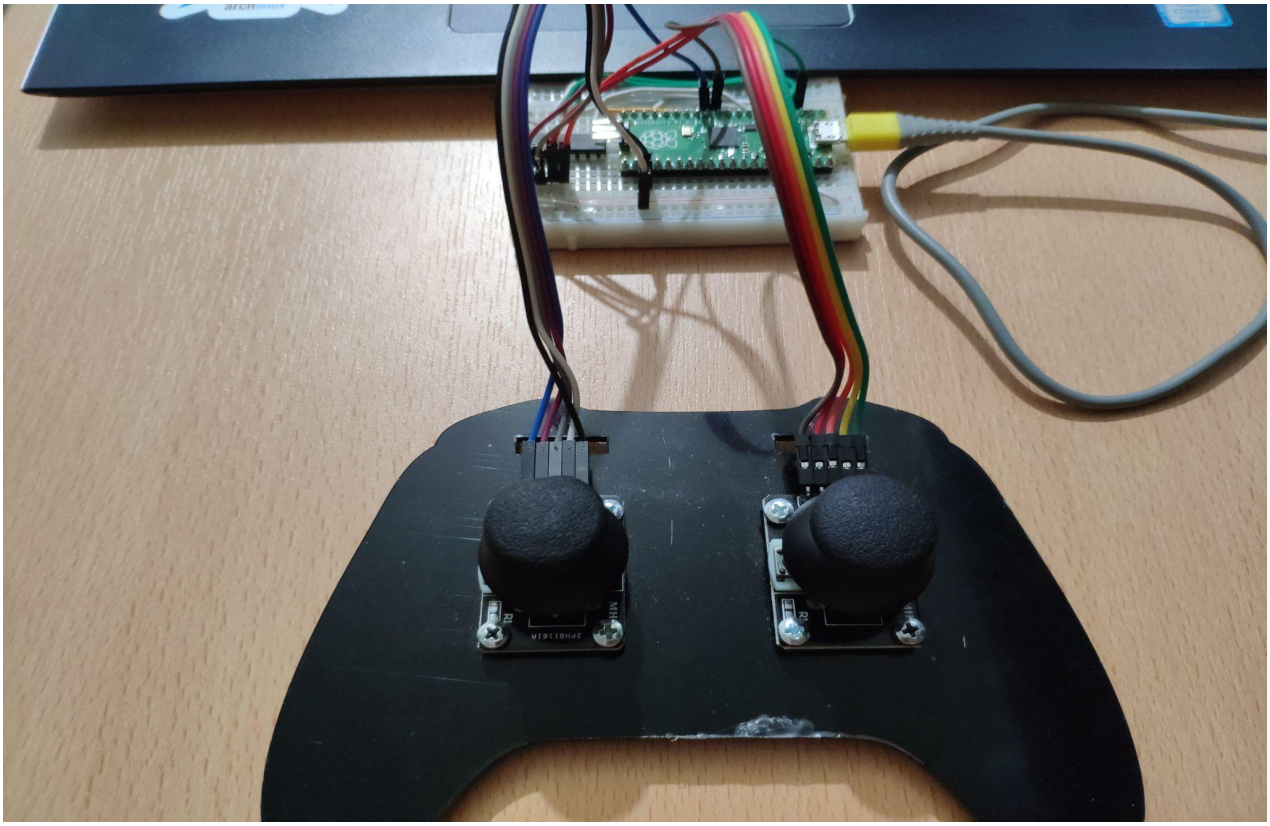
We are using Raspberry Pi Pico as controller, two Joysticks and a button(Fig 3.1) to get input from the user tiny usb interface to appear as a keyboard and determine the directions according to joystick movements and buttons. We used MCP-3208 integrated circuit which has 8 ADC (Analog Digital Converter) channels to get analog and convert those analog values that are coming from the joystick to 12 bit digital values. We also have two buttons for the joystick.

Outputs:

3-axis control for drone inside simulation.



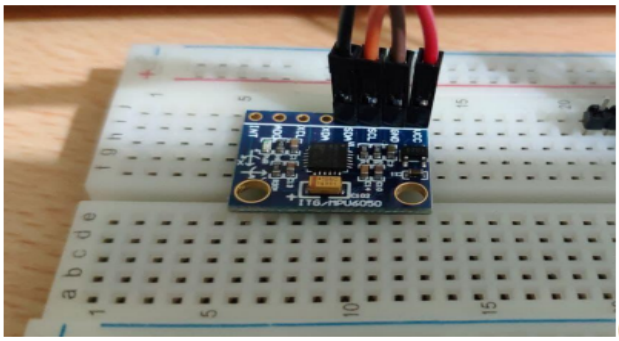
FINAL VERSION OF GAMEPAD

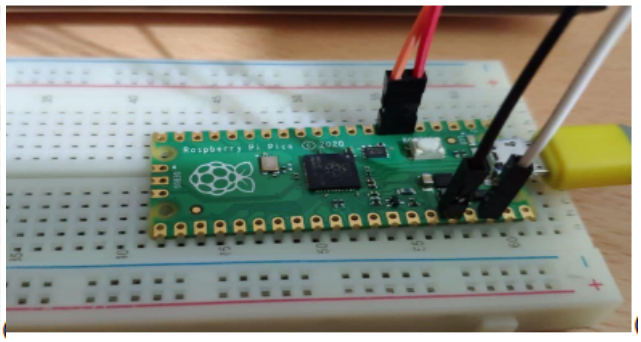


4.1.4 Glove Controller

The glove uses a gyroscope/accelerometer sensor to sense the motion of the wrist and flexible sensors for movement of the finger. Hand actions are used as a move and finger actions are used as a command, transmitted by the glove to the simulator unit using the serial communication port.

We are using Raspberry Pi Pico as a controller(Fig 4.1) and MPU6050 gyroscope/ accelerometer(Fig 4.2) sensor to recognize the state of hand and we can get data by using human interface device We are calculating the x-y-z axis to determine directions, we determine drone ascend and descend behaviour by using flex sensor.



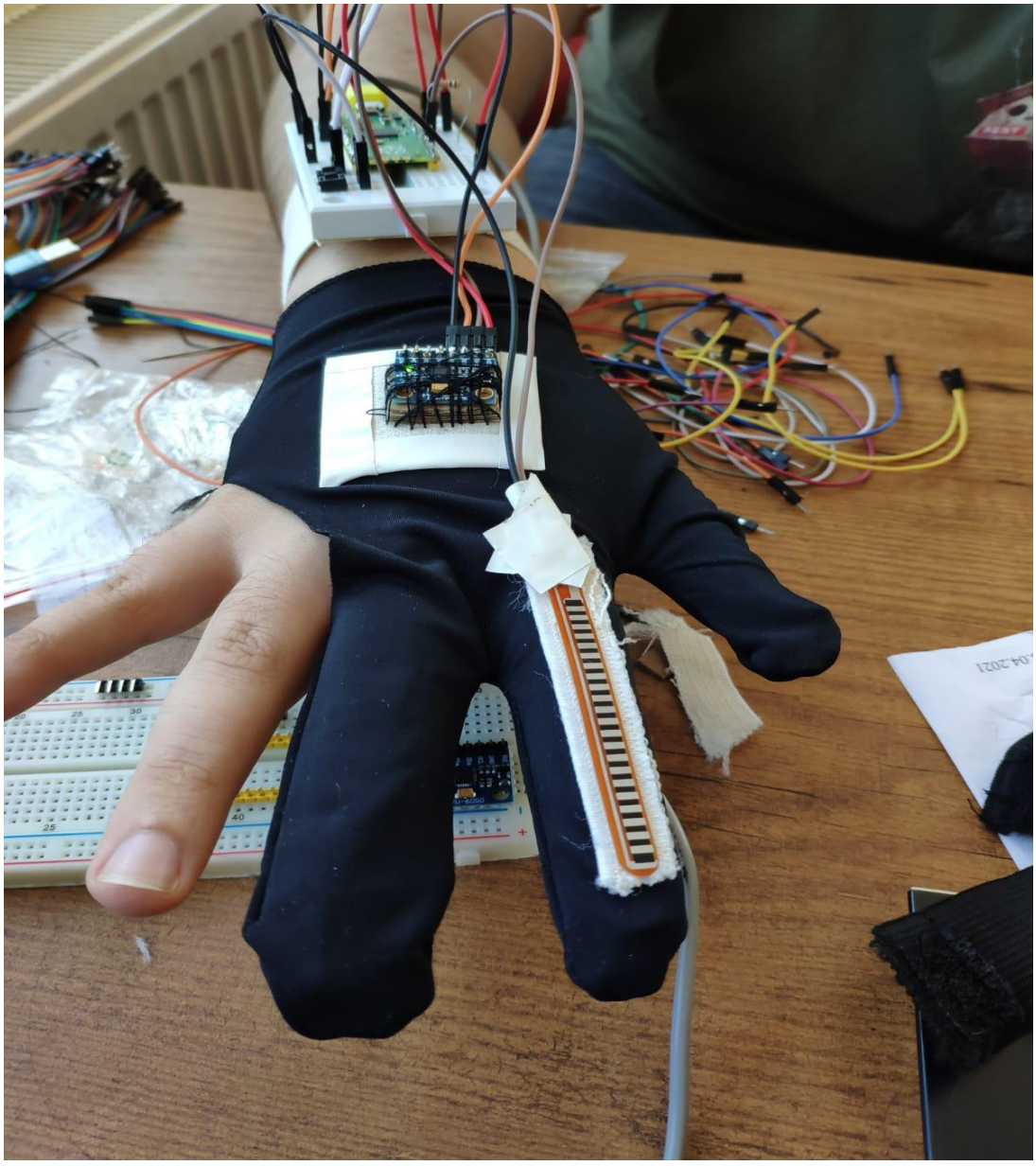


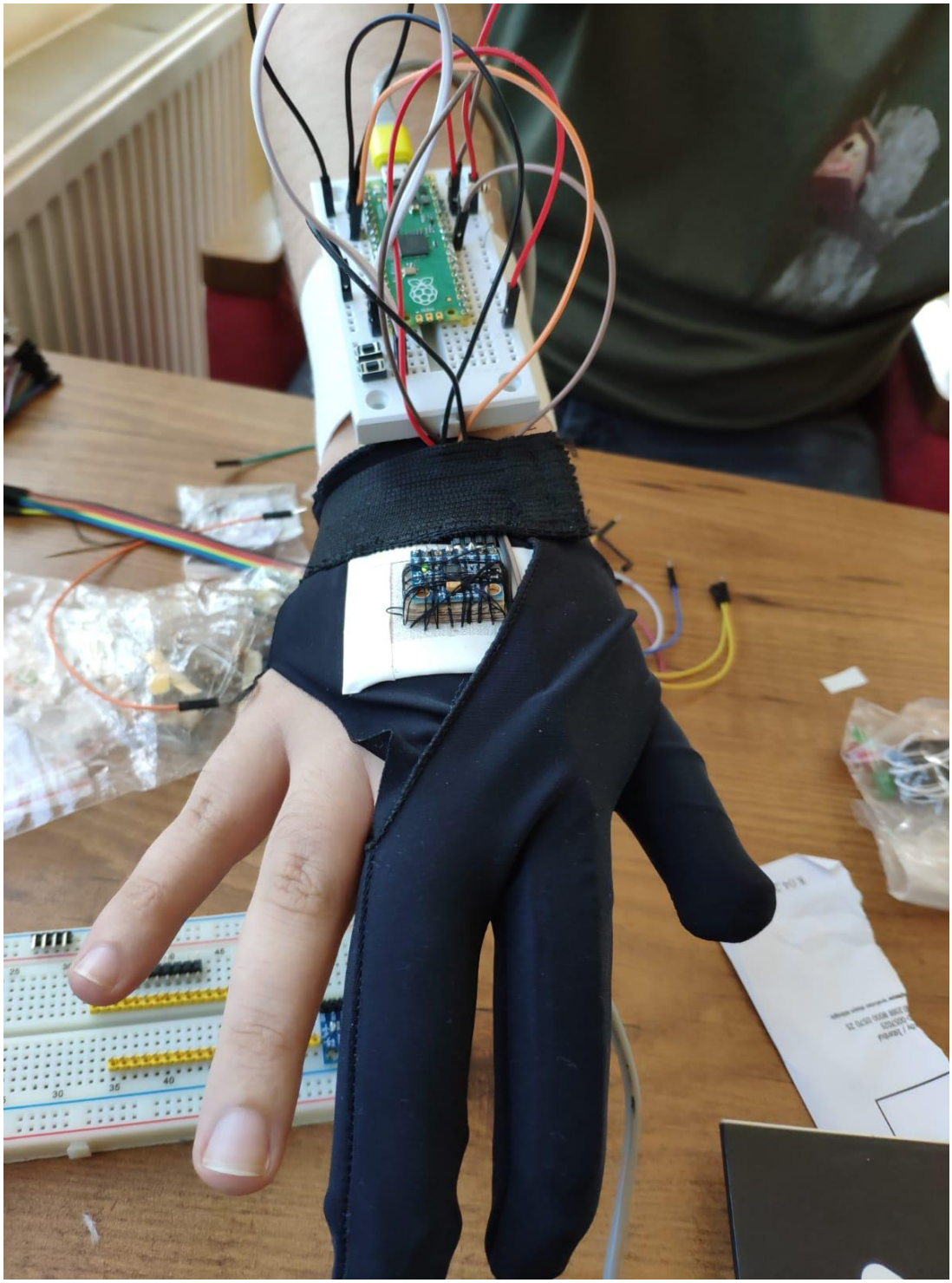
We decided to buy hardware materials for each team member thus incase of lockdown we can develop the hardware at home so it should be cheap. Firstly we bought a stm32f401 microcontroller but we can not use it. Since it's pretty new hardware it is hard to find a solution in case of any error. So then we decided to use a raspberry pi pico with its user friendly manual.

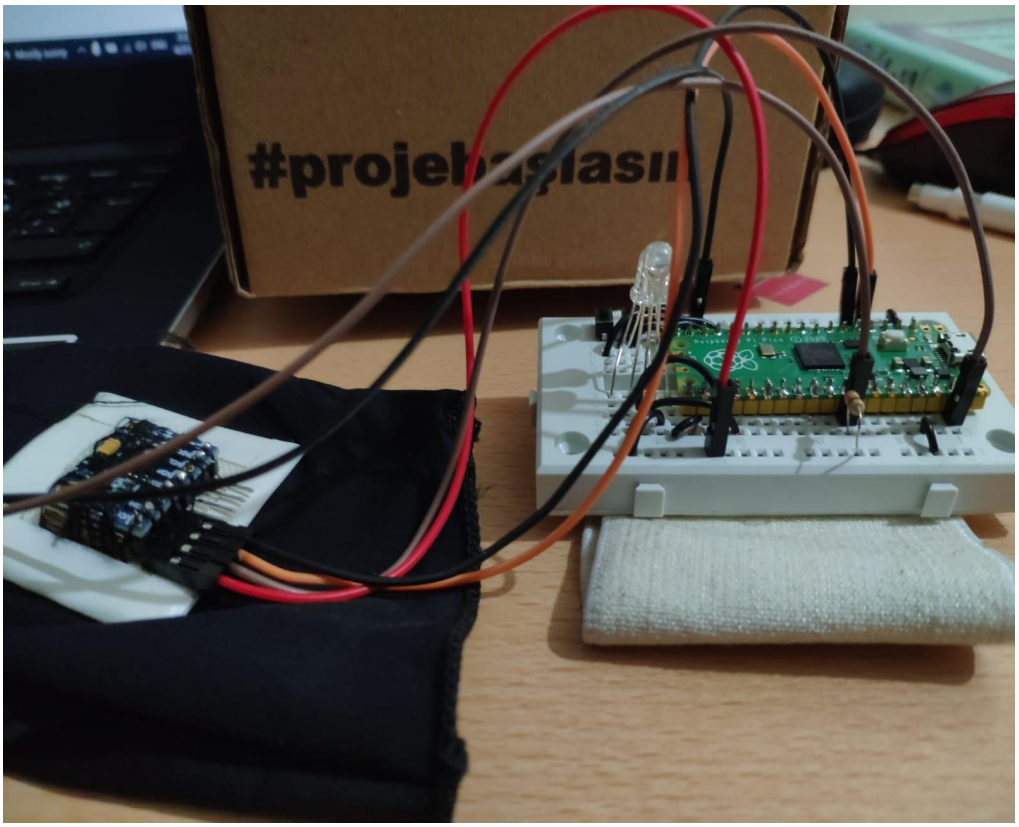
Outputs:

Glove controller that provides 3-axis control of the drone via hand.

FINAL VERSION OF GLOVE







4.1.5 Communication Of Integration

We have to pass the digital value read by glove or gamepad to the simulation environment. This module will work on how to process input data, how to send it using controller's ports and use it as commands on simulation environments.

Requirements:

- Glove Controller
- Gamepad Controller
- Flash drive and necessary programs(e.g. simulation software)

Outputs:

A realistic drone simulation that is controlled by a glove controller or gamepad controller in GTU Campus Map.

4.2 Implementation Of Modules

4.2.1 Simulation Modelling

We replicated a portion of the GTU campus. We used free assets from the internet and we created some basic ones of our own using Blender. None of the members in our group is skilled enough to create assets that reflect our buildings exactly so we had to make do with what we could find online and the basic assets that we created. Using those assets we were able to somewhat replicate the Marmaray line, most of the buildings in our university(excluding Materials Engineering department building, the new conference building and a few more). We also did some landscaping and tried to reflect the nature of our campus accurately. Unfortunately running the simulator from an USB stick makes it a bit laggy if we try to make the plant population accurate so we had to decrease the number of plants present in the map. We made sure that the bottleneck causing this problem was running it from the USB stick. So because of this we had to cut back on graphics for the sake of performance.

4.2.2 Simulation Kinematics

Our drone consists of 4 motors and it uses a PID algorithm. In every single frame, according to input's direction, it calculates a 3-dimensional vector and clamps this value with minimum and maximum speed of our drone. Every motor applies this force to the drone and it saves the calculated value as the last force applied for the further use. After that, our PID algorithm starts to work. For every drone direction (Altitude, Pitch, Roll and Yaw), firstly, it calculates what distance the drone should have gone after the last forces it's applied it will be our error value. According to how many seconds have passed from the last error, it calculates the integral and sums it with the last integral. And then, it calculates what output force to correct the error of the drone with PID gain values (x = 0.2, y = 0.03, z = 0.05). Calculated error output force is applied to every motor and this loop runs until the game ends.

4.2.3 Gamepad Controller

We used 2 analog sticks with a button inside them. We used Raspberry Pi Pico for microcontroller which is enough for our case and MCP-3208 for Analog Digital Converter which is for analog sticks.

Since Raspberry Pi Pico has only 3 ADC channels and an analog stick consumes 2 ADC channels, We had to use an external ADC which is MCP-3208. MCP-3208 has 8 channel ADC and has a 12 bit sampling rate.

In order to communicate with MCP-3208, We used the SPI (Serial Peripheral Interface) protocol. We communicate with the peripheral device using an SPI port that is inside the raspberry pi pico. And determine which device is MOSI (Master output, slave input) and MISO (Master input, slave output). In our case MOSI is raspberry pi pico and MISO is MCP-3208. After that, using the Chip Select command, We read the channels inside the MCP-3208 and convert those values to 12 bit digital values.

Left analog stick controls the orientation of the drone like moving the camera right or left and ascending the drone or descending the drone.

Right analog stick moves the drone in the landscape like moving forward, backward, right or left.

Left button throttles the drone and Right button restarts the simulation.

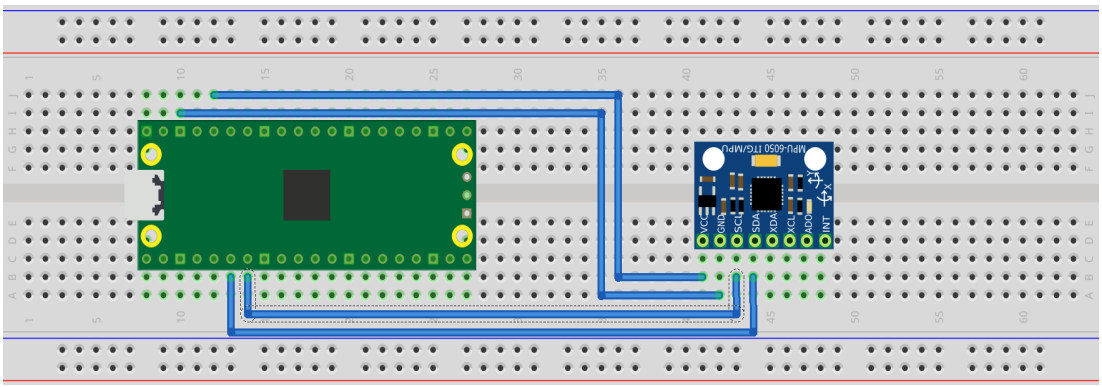
Joystick is powered by 3.3V source and gets the power source from raspberry pico's internal power output pin.

4.2.4 Glove Controller

We used the gyroscope and accelerometer sensor to manually control the drone. We made sense of the data coming from this sensor and turned it into 6 directions (forward, backward, right, right lean, left, left lean). In this way, the user will be able to move the drone in the desired direction with his hand. We used the flex sensor to raise and lower the drone. The flex sensor on the index finger gives us the data whether the finger is squeezed or not. If the data from the flex sensor is in the direction of squeezing the finger, the drone takes off or descends. There are 3 modes

(reset, ascend, descend) for this. This mode is changed by the button. In the initial state, the mode is in reset state and the LED light turns blue to indicate this. When the button is pressed once, the rise mode is entered and the green light turns on to indicate this. Pressing the button again will switch to descent mode and the red light will illuminate to indicate this. To put it back into reset mode, the reset button, which is a separate button, must be pressed and the drone turns into blue light mode.

Only recovers raw data from the sensor. There are various calibration options available that should be used to ensure that the final results are accurate. It is also possible to wire up the interrupt pin to a GPIO and read data only when it is ready.Wiring up the device requires 4 jumpers, to connect VCC (3.3v), GND, SDA and SCL. The example here uses I2C port 0, which is assigned to GPIO 4 (SDA) and 5 (SCL) in software. Power is supplied from the 3.3V pin.



The adc pin was used for the Flex sensor, the data from the sensor was evaluated within a certain range. In this way, it is understood whether the finger is squeezed or not.

GPIO pins are used for buttons and leds, so data is sent and received.Sensors are powered by 3.3V source and get the power source from raspberry pico’s internal power output pin.

4.2.5 Communication and Integration

In order to communicate with the simulation, We first determined to use Serial Port, but We had some issues related to the Serial Port which are;

- Not opening a tty that reads from serial port before the simulation, made simulation unable to read the Serial Port.
- Unplugging the device and plugging it again, made the simulation lose communication with the device.
- Only one Serial Communication could be achieved on the same machine. So we couldn't read data from the joystick and the glove at the same time.
- It had port number problems where the port number could vary on different computers.

Because of these issues we have given up on Serial Port Communication. Instead we chose to make our hardwares as a keyboard using the "tinyUSB" library from pico-SDK.

In order to establish this, we declared our keyboard configuration using HID key commands from the pico standard library. And set those key commands appropriately with respect to the input we received from the hardware. Lastly we send those key Command values or conditions to the tiny USB library function (tud_hid_keyboard_report())to fulfill this action.

5. SETUP AND RUN

Quick Pico Setup:

You can get this script by running the following command in a terminal:

```
wget https://raw.githubusercontent.com/raspberrypi/pico-setup/master/pico_setup.sh
```

Then make the script executable with:

```
chmod +x pico_setup.sh
```

and run it with,

```
./pico_setup.sh
```

Once it has run, you will need to reboot your Raspberry Pi,

```
sudo reboot
```

Get SDK:

Development on the board is fully supported with both a C/C++ SDK. We talk about how to get started with the SDK, and walk you through how to build, install, and work with the SDK toolchain.

```
cd ~/
mkdir pico
cd pico
```

Then clone the pico-sdk git repositories:

```
git clone -b master https://github.com/raspberrypi/pico-sdk.git
cd pico-sdk
git submodule update --init
```

Install The Toolchain:

To build the applications in source codes, you'll need to install some extra tools. To build projects you'll need CMake, a cross-platform tool used to build the software, and the GNU Embedded Toolchain for Arm. You can install both these via apt from the command line.

```
sudo apt update
sudo apt install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi build-essential
```

Updating the SDK:

```
cd pico-sdk /
git pull
git submodule update
```

Build:

From the pico directory we created earlier, cd into <pico_project> and create a build directory.

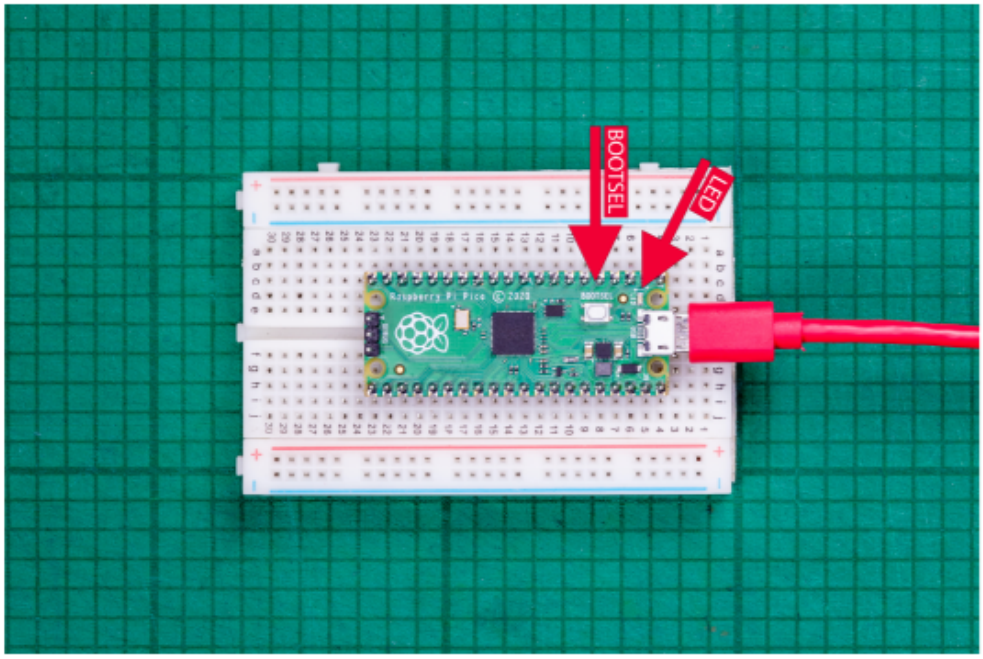
```
cd <pico_project>
mkdir build
cd build
export PICO_SDK_PATH=../../pico-sdk
cmake ..
make
```

Load & Run:

The fastest method to load software onto a RP2040-based board for the first time is by mounting it as a USB Mass Storage Device. Doing this allows you to drag a file onto the board to program the flash. Go ahead and connect the Raspberry Pi Pico to your Raspberry Pi using a micro-USB cable, making sure that you hold down the BOOTSEL button as you do so, to force it into USB Mass Storage Mode.

If you are running the Raspberry Pi Desktop the Raspberry Pi Pico should automatically mount as a USB Mass Storage Device. From here, you can Drag-and-drop <project_file>.uf2 onto the Mass Storage Device. RP2040 will reboot, unmounting itself as a Mass Storage Device, and start to run the flashed code, see this figure.

Alternative Raspberry Pi Pico project building is also explained in this video: https://www.youtube.com/watch?v=NCaL6tXAF0c&t=484s&ab_channel=GaryExplains



All functions provided by Pico SDK were used in this project. CMake file in the desired format has been prepared.

To run the simulation, the executable file in the project file must be run.

6. CONCLUSION

For simulation modeling module:

We have replicated the GTU campus partially. The graphics are a bit on the weak side but as we explained the implementation part, it is caused by the USB bottleneck. Other than that we kind of achieved what we aimed for in this module.

For simulation kinematics module:

We have a drone which satisfies the requirements we specified in the module description. Our drone can interact with the world, has a top speed and can stay in the air while no input is present.

For Gamepad Controller module:

We have built a gamepad controller which has 2 joysticks and 2 buttons. We can use it to control our drone just as we specified in the requirements.

For Glove Controller module:

We have built a glove controller which produces inputs according to our hand and finger movements as we specified in the requirements. We can successfully use it to control our drone.

For Communication and Integration module:

In this module we aimed to have a simulation that can run from a bootable USB drive automatically and run our hardwares in our simulation running on that USB.

We have achieved this successfully as we have a USB that has our simulation in it, it is bootable and our hardware works with it.

7. MEMBERS OF MODULES

BERKE BELGİN	Communication and Integration Gamepad Controller Glove Controller
EZGİ KOTAN	Gamepad Controller Glove Controller Communication and Integration
İLKAN MERT OKUL	Communication and Integration Gamepad Controller Glove Controller
KAMİL KAYA	Communication and Integration Gamepad Controller Glove Controller
FURKAN ÖZEV	Communication and Integration Gamepad Controller Glove Controller
YUSUF AKGÜL	Communication and Integration Gamepad Controller Glove Controller
BARAN HASAN BOZDUMAN	Communication and Integration Gamepad Controller Glove Controller
MUZAFFER DEHA PELİT	Communication and Integration Gamepad Controller Glove Controller
CANBERK ARICI	Simulation Modelling Simulation Kinematics Communication and Integration
TÜRKER TEZCAN	Simulation Modelling Simulation Kinematics Communication and Integration
ABDULLAH KÜSGÜLÜ	Simulation Modelling

	Simulation Kinematics Communication and Integration
ZAFER ALTAY	Gamepad Controller Glove Controller Communication and Integration
ALİNA KURALOVA	Simulation Modelling Simulation Kinematics Communication and Integration