Bachelor of Engineering in Robotics and Artificial Intelligence

Team Project Module

Project Title:

AlphaMini – Robot Classroom Assistant Teacher (Rcat)

Group Progress Report 3

Name
Group Leader: Monish Naresh Kumar
Member #1: Cornelius Fulvian
Member #2: La Min Aung
Member #3: Joseph Santhosh Aaron

Project Supervisor: Sarina Tajudin

Date: 26/10/2025

Words Count:

(excl. Table of content/references)

Abstract

For this module we have been challenged to explore and use the AlphaMini, an intelligent humanoid robot developed by UBTECH Robotics. The AlphaMini is baked with rich artificial intelligence hardware, combined with uCode graphical programming tools and curriculum, to let students design, build, program scenarios with AI technologies, make the robot respond to human’s directions and environmental changes, interact with self-built machineries, and actualize other brilliant ideas. (NIE, NTU , n.d.)

Table of Contents

List of Figures

List of Tables

Table #	Table Label	Table Page
Table 1:	Milestones	7
Table 2:	Incomplete Tasks	9
Table 3:	Role Assignments	10
Table 4:	Component to Microcontroller Connections Guide	10
Table 5:	Code for clock with Alarm and Timer functionalities	11
Table 6:	Code for Navigation – Manual control	13
Table 7:	Code for Navigation – Automation	15
Table 8:	Code for Navigation – Obstacle avoidance	17
Table 9:	Code for Attendance Taking System – AlphaMini side	22
Table 10:	Code for Attendance Taking System – Remote PC side	25
Table 11:	Members' Signatures	28

1. Introduction

1.1 Updated Aim and Objectives

Aim

To reinstate, the fundamental aim of this project is to implement the AlphaMini as a **Robot Classroom Assistant Teacher (Rcat)** designed to support the main teacher in managing and automating mundane tasks and improve efficiency of classroom. Our Rcat is therefore capable of reducing teacher's workload on some admin matters (attendance, in class quiz handling, one to one language guide) in turn demonstrate its capabilities to improve and enrich the efficiency of the learning environment for both students and teachers/school admin staff.

Objectives

The objectives of this project are as follows:

1) Compact attendance taking system - Completed

- The AlphaMini must be capable of identifying a QR code is presented to it
- The AlphaMini must capture an image of the QR code and send it to Remote PC
- Remote PC should decipher the QR code
- Data from the QR code like date time and name should be logged in an external SQL / Excel Sheet.

2) Navigation around classroom - Completed

- Basic Wandering
- Obstacle Avoidance
- Point to point movement

3) Timekeeper - In Progress

- Design and Build a circuit with ESP32 with a LCD / 7 segment display for a timeclock
- Program the AlphaMini to be able to start stop and set the timeclock
- Enable voice activation eg; "Start a timer for a 30 min test"

4) Peripherals – In Progress

- 3D Print a Idle Hub / Home station for the Rcat to charge and sync to the wifi
- 3D Print a Teachers Stick for aesthetic and potential realism movements
- 3D Print a mock small scale classroom environment for simulation

5) Sensing and object recognition - Incomplete

- Train an object recognition logic model
- Install Object Identification in Alphamini

6) Language Teaching guide - Incomplete

- The Alphamini should be capable of Speech recognition
- Link the Alphamini to a translation software or AI tool for language help
- Reply to the user with correct help in language

7) Interactive quiz handler - Incomplete

- Not finalized on execution but should be able to conduct a quiz in real time.

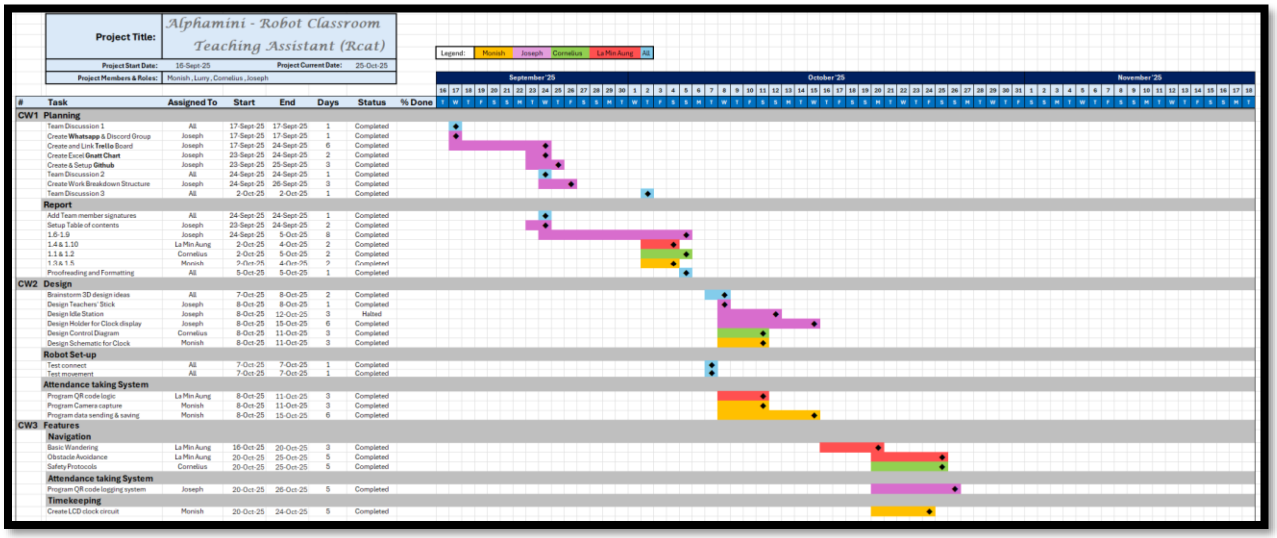
1.2 Current State of the project

Update your current state of the project here.

Milestones	Date of Completion	Current Status	Comments
Compact attendance taking system			
1) Implement and Test QR code scanning	**11-10-25**	**-**	Captures QR code and sends to remote PC
2) Implement QR code logging to an online database	**24-10-25**	**-**	Deciphers QR code and logs data in SQL
3) Integrate both together	**25-10-25**		
3D Prints			
1) Finalize on 3D printed object to enhance the Rcat's functionality	**13-10-25**	Idle Hub still TBC	Peripherals: Table, chair, stick, clock holder
2) Design and Print	**14-10-25**	**-**	One set done
3) Install and Test functionality	**16-10-25**	**-**	-
Final Robot Functionality			
1) Robot Control Diagram	**15-10-25**	**-**	Changes made accordingly to code variances
Timekeeper			
1) Setup time logic and tracker	**22-10-25**	**-**	-
2) Implement and Test time management features	**25-10-25**	**-**	-
3) Real time trial run in possible scenarios	**26-10-25**	-	Minor improvements pending
Navigation			
1) Basic Wandering	**20-10-25**	**-**	-
2) Obstacle Avoidance	**22-10-25**	**-**	-
3) Mapping & Path planning	**26-10-25**	**-**	Template only
4) Safety	**26-10-25**	**-**	

Table 1: Milestones

1.3 Updated Schedule



Project Title:	Alphamini - Robot Classroom Teaching Assistant (Rcat)
Project Start Date:	16-Sept-25
Project Current Date:	25-Oct-25
Project Members & Roles:	Monish , Lurry , Cornelius , Joseph

Legend: Monish, Joseph, Cornelius, La Min Aung, All

#	Task	Assigned To	Start	End	Days	Status	% Done
CW1	**Planning**						
	Team Discussion 1	All	17-Sept-25	17-Sept-25	1	Completed	
	Create **Whatsapp** & Discord Group	Joseph	17-Sept-25	17-Sept-25	1	Completed	
	Create and Link **Trello** Board	Joseph	17-Sept-25	24-Sept-25	6	Completed	
	Create Excel **Gnatt Chart**	Joseph	23-Sept-25	24-Sept-25	2	Completed	
	Create & Setup **Github**	Joseph	23-Sept-25	25-Sept-25	3	Completed	
	Team Discussion 2	All	24-Sept-25	24-Sept-25	1	Completed	
	Create Work Breakdown Structure	Joseph	24-Sept-25	26-Sept-25	3	Completed	
	Team Discussion 3	All	2-Oct-25	2-Oct-25	1	Completed	
	Report						
	Add Team member signatures	All	24-Sept-25	24-Sept-25	1	Completed	
	Setup Table of contents	Joseph	23-Sept-25	24-Sept-25	2	Completed	
	1.6-1.9	Joseph	24-Sept-25	5-Oct-25	8	Completed	
	1.4 & 1.10	La Min Aung	2-Oct-25	4-Oct-25	2	Completed	
	1.1 & 1.2	Cornelius	2-Oct-25	5-Oct-25	2	Completed	
	1.3 & 1.5	Monish	2-Oct-25	4-Oct-25	2	Completed	
	Proofreading and Formatting	All	5-Oct-25	5-Oct-25	1	Completed	
CW2	**Design**						
	Brainstorm 3D design ideas	All	7-Oct-25	8-Oct-25	2	Completed	
	Design Teachers' Stick	Joseph	8-Oct-25	8-Oct-25	1	Completed	
	Design Idle Station	Joseph	8-Oct-25	12-Oct-25	3	Halted	
	Design Holder for Clock display	Joseph	8-Oct-25	15-Oct-25	6	Completed	
	Design Control Diagram	Cornelius	8-Oct-25	11-Oct-25	3	Completed	
	Design Schematic for Clock	Monish	8-Oct-25	11-Oct-25	3	Completed	
	Robot Set-up						
	Test connect	All	7-Oct-25	7-Oct-25	1	Completed	
	Test movement	All	7-Oct-25	7-Oct-25	1	Completed	
	Attendance taking System						
	Program QR code logic	La Min Aung	8-Oct-25	11-Oct-25	3	Completed	
	Program Camera capture	Monish	8-Oct-25	11-Oct-25	3	Completed	
	Program data sending & saving	Monish	8-Oct-25	15-Oct-25	6	Completed	
CW3	**Features**						
	Navigation						
	Basic Wandering	La Min Aung	16-Oct-25	20-Oct-25	3	Completed	
	Obstacle Avoidance	La Min Aung	20-Oct-25	25-Oct-25	5	Completed	
	Safety Protocols	Cornelius	20-Oct-25	25-Oct-25	5	Completed	
	Attendance taking System						
	Program QR code logging system	Joseph	20-Oct-25	26-Oct-25	5	Completed	
	Timekeeping						
	Create LCD clock circuit	Monish	20-Oct-25	24-Oct-25	5	Completed	

Figure 1: Gnatt Chart

Work Breakdown structure of work -

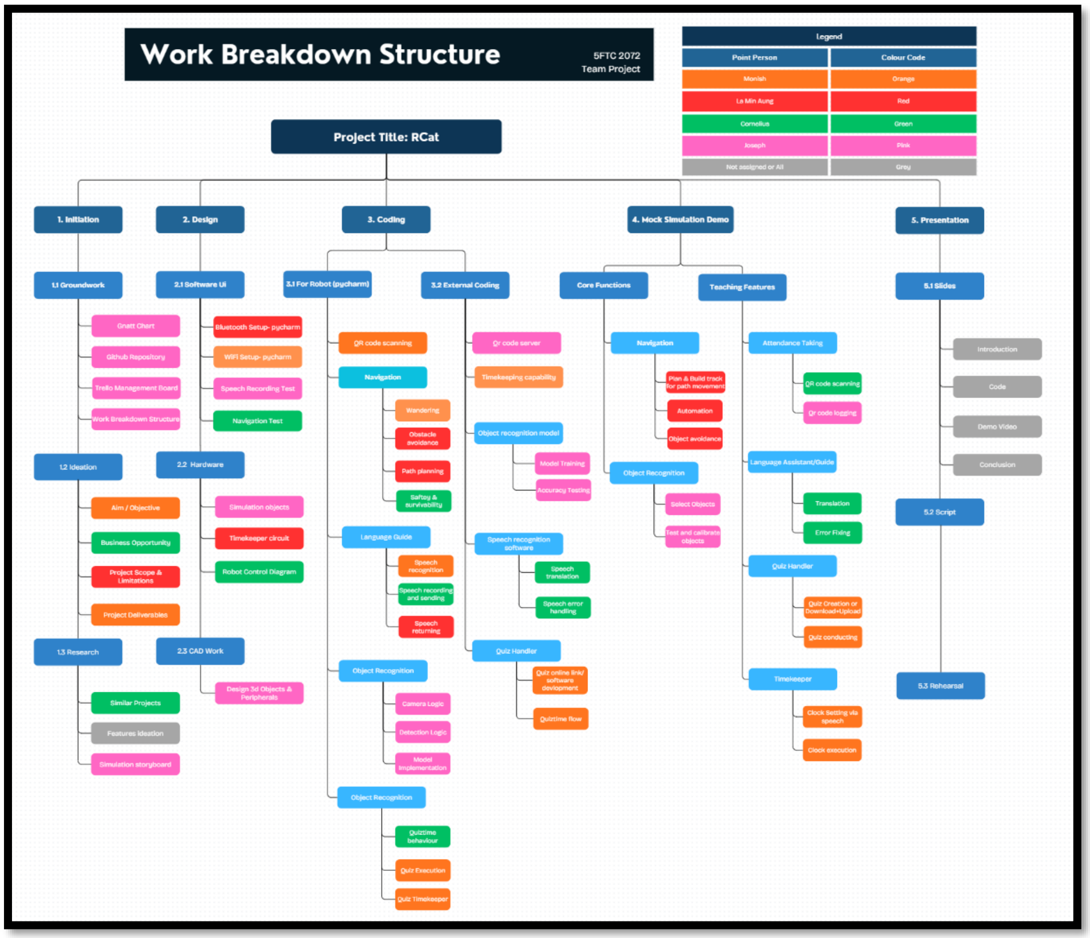


Figure 2: Updated Work Breakdown Structure

The table below lists the deliverables yet to be met -

Incomplete Tasks	Assignment	Current Status
Navigation		
Mapping	**La Min Aung**	**Planning**
Localization	**Monish**	**Planning**
Path Planning	**Joseph**	**Ideation**
Calibration & Fine Tuning	**Cornelius**	**Backlog**
Language Guide (speech recognition)		
Speech recognition	**Monish**	**Ideation**
Speech translation	**La Min Aung**	**Ideation**
Speech error-correction support	**La Min Aung**	**Backlog**
Add Timekeeper support	**Monish**	**Planning**
Interactive quiz maker & handler		
Quiz making	**Cornelius**	**Backlog**
Quiz handling	**Cornelius**	**Backlog**
Sensing and object recognition		
Create object recognition model	**Joseph**	**Backlog**
object recognition (static)	**Monish**	**Backlog**
object recognition (dynamic)	**La Min Aung**	**Backlog**
Evaluation		
Integrate all the functions together and ensure all functions are acting sensibly	**ALL**	**NA**
Create Simulation layer	**ALL**	**NA**
Storyboard final presentation	**Joseph**	**Planning**
Troubleshooting and bug fixing	**ALL**	**NA**

Table 2: Incomplete Tasks

2. Project Updates

2.1 Individual Contributions

We stayed primarily faithful to the roles assigned previously to ensure even workload distribution and to attenuate member's strengths, but since we have completed project research and 3d printing designs some roles have been reformatted.

Role class	Monish	La Min Aung	Cornelius	Joseph
Primary	Coding	Coding	Coding	Groundwork
Secondary	Building	Building	Building	Coding

Table 3: Role Assignments

Student#1 - Monish Naresh Kumar
Individual Aim (till 26/10/2025):

For this period of coursework, I was responsible to implement the "timekeeper" physically and create a working prototype, also developing and kickstarting the navigation/movement of Alpha Mini. The plan for next steps are to be finalized by 26/10/25, finding how to move forward with existing limitations.

I) Timekeeper

As the previous report mentioned on implementing the "timekeeper" as a physical separate clock, the task was successfully done. The clock uses 3 components, the ESP8266 (ESP32, as previously mentioned, was not used as it's not given in lab for this module and I used my personal ESP8266 Module) a LCD display, and a buzzer for alarm purposes. The connections were simple; the below table shows the connection summary.

ESP8266	components
3.3V	LCD VCC
GND	LCD GND & BUZZER GND
D1	LCD SCL
D2	LCD SDA
D5	BUZZER POSITIVE

Table 4: Component to Microcontroller Connections Guide

For now, the LCD displays the time in HH:MM:SS format and date in DD/MM/YYYY according to Singapore. The date and time are retrieved through Wi-Fi. This is possible because the microcontroller, ESP8266, has inbuilt Wi-Fi module.

A predefined timer can be set in the code; this is the time which the microcontroller constantly compares with the current time. When both time matches it enables pin D5 so the buzzer starts working, this acts like an alarm. The prototype is made in such a way that timer can be set in code only because the time should be set by AlphaMini, and this version is just for testing purposes.

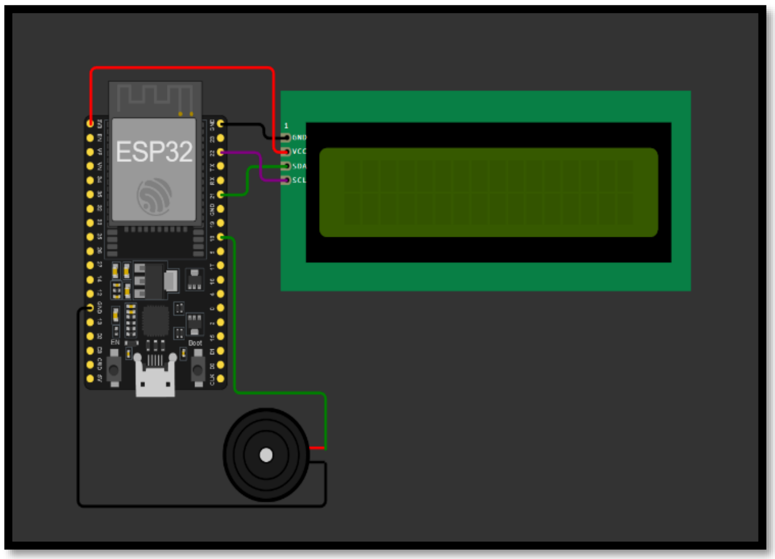
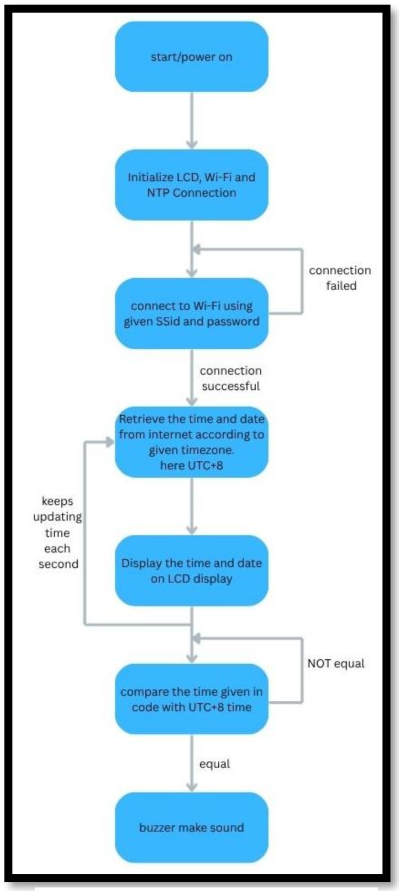


Figure 5: Clock Circuit Diagram

Referring to the figure 3 the "keeps updating time each second" segment, does not affect other segments but is primarily ensuring the accuracy of time, every loop.

Even if the Wi-Fi is disconnected, the clock still keeps count but can be prone to very minor inaccuracies after prolonged periods of time offline.


start/power on
Initialize LCD, Wi-Fi and NTP Connection
connect to Wi-Fi using given SSid and password
connection failed
connection successful
Retrieve the time and date from internet according to given timezone. here UTC+8
keeps updating time each second
Display the time and date on LCD display
compare the time given in code with UTC+8 time
NOT equal
equal
buzzer make sound
Figure 4: Clock Logic Flowchart

<u>Code for clock with Alarm and Timer functonalities (based on flowchart Figure 4)</u>

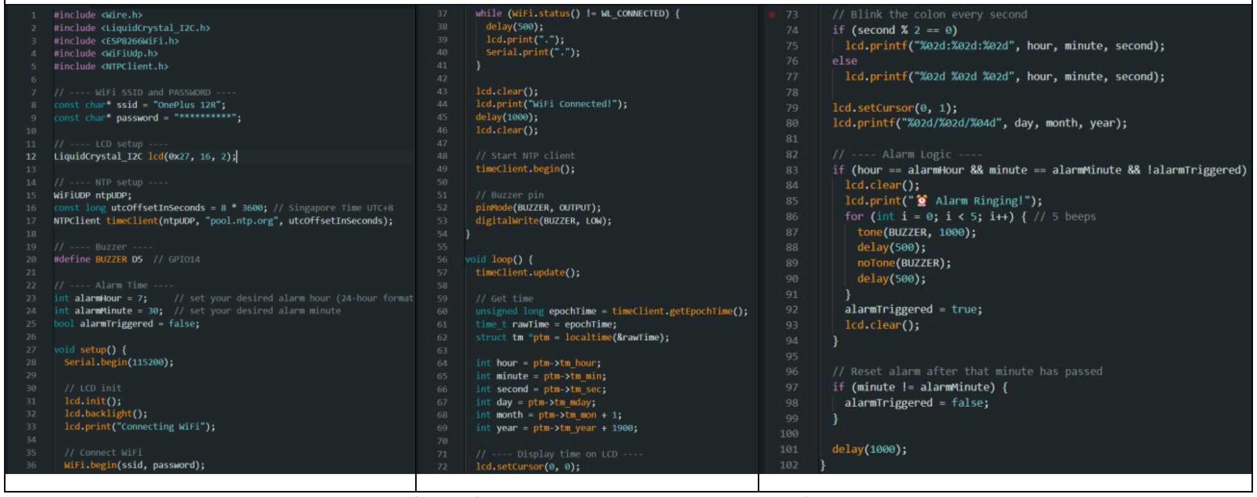

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <NTPClient.h>

// ---- WiFi SSID and PASSWORD ----
const char* ssid = "OnePlus 12R";
const char* password = "**********";

// ---- LCD setup ----
LiquidCrystal_I2C lcd(0x27, 16, 2);

// ---- NTP setup ----
WiFiUDP ntpUDP;
const long utcOffsetInSeconds = 8 * 3600; // Singapore Time UTC+8
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);

// ---- Buzzer ----
#define BUZZER D5  // GPIO14

// ---- Alarm Time ----
int alarmHour = 7;     // set your desired alarm hour (24-hour format
int alarmMinute = 30;  // set your desired alarm minute
bool alarmTriggered = false;

void setup() {
  Serial.begin(115200);

  // LCD init
  lcd.init();
  lcd.backlight();
  lcd.print("Connecting WiFi");

  // Connect WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    lcd.print(".");
    Serial.print(".");
  }

  lcd.clear();
  lcd.print("WiFi Connected!");
  delay(1000);
  lcd.clear();

  // Start NTP client
  timeClient.begin();

  // Buzzer pin
  pinMode(BUZZER, OUTPUT);
  digitalWrite(BUZZER, LOW);
}

void loop() {
  timeClient.update();

  // Get time
  unsigned long epochTime = timeClient.getEpochTime();
  time_t rawTime = epochTime;
  struct tm *ptm = localtime(&rawTime);

  int hour = ptm->tm_hour;
  int minute = ptm->tm_min;
  int second = ptm->tm_sec;
  int day = ptm->tm_mday;
  int month = ptm->tm_mon + 1;
  int year = ptm->tm_year + 1900;

  // ---- Display time on LCD ----
  lcd.setCursor(0, 0);
  // Blink the colon every second
  if (second % 2 == 0)
    lcd.printf("%02d:%02d:%02d", hour, minute, second);
  else
    lcd.printf("%02d %02d %02d", hour, minute, second);

  lcd.setCursor(0, 1);
  lcd.printf("%02d/%02d/%04d", day, month, year);

  // ---- Alarm Logic ----
  if (hour == alarmHour && minute == alarmMinute && !alarmTriggered)
    lcd.clear();
    lcd.print("⏰ Alarm Ringing!");
    for (int i = 0; i < 5; i++) { // 5 beeps
      tone(BUZZER, 1000);
      delay(500);
      noTone(BUZZER);
      delay(500);
    }
    alarmTriggered = true;
    lcd.clear();
  }

  // Reset alarm after that minute has passed
  if (minute != alarmMinute) {
    alarmTriggered = false;
  }

  delay(1000);
}
```

Table 5: Code for clock with Alarm and Timer functionalities

II) Navigation (manual control)

Before starting to code on the AlphaMini, we have to test the connection between the robot and the remote computer. This is done by connecting both the robot and remote computer to the same Wi-Fi and then searching for the robot using its number, in our case the number is “00213”. After connecting the robot, we proceed with the motor functions.

For navigation first I set up manual control of the AlphaMini by programming the use of the keyboard of the remote computer. I have made four motor functions + two extra functions. The motor functions include *move forward*, *move backwards*, *turn right* and *turn left*. The extra functions are *raise hand* and *exit manual control*. All the motor functions operate with a step size of 1, that is whatever the robot is said to do, it does 1*function. For example, if move forward is given then the robot moves 1 step forward.

The flowchart below shows the logic and workflow of the manual control function.

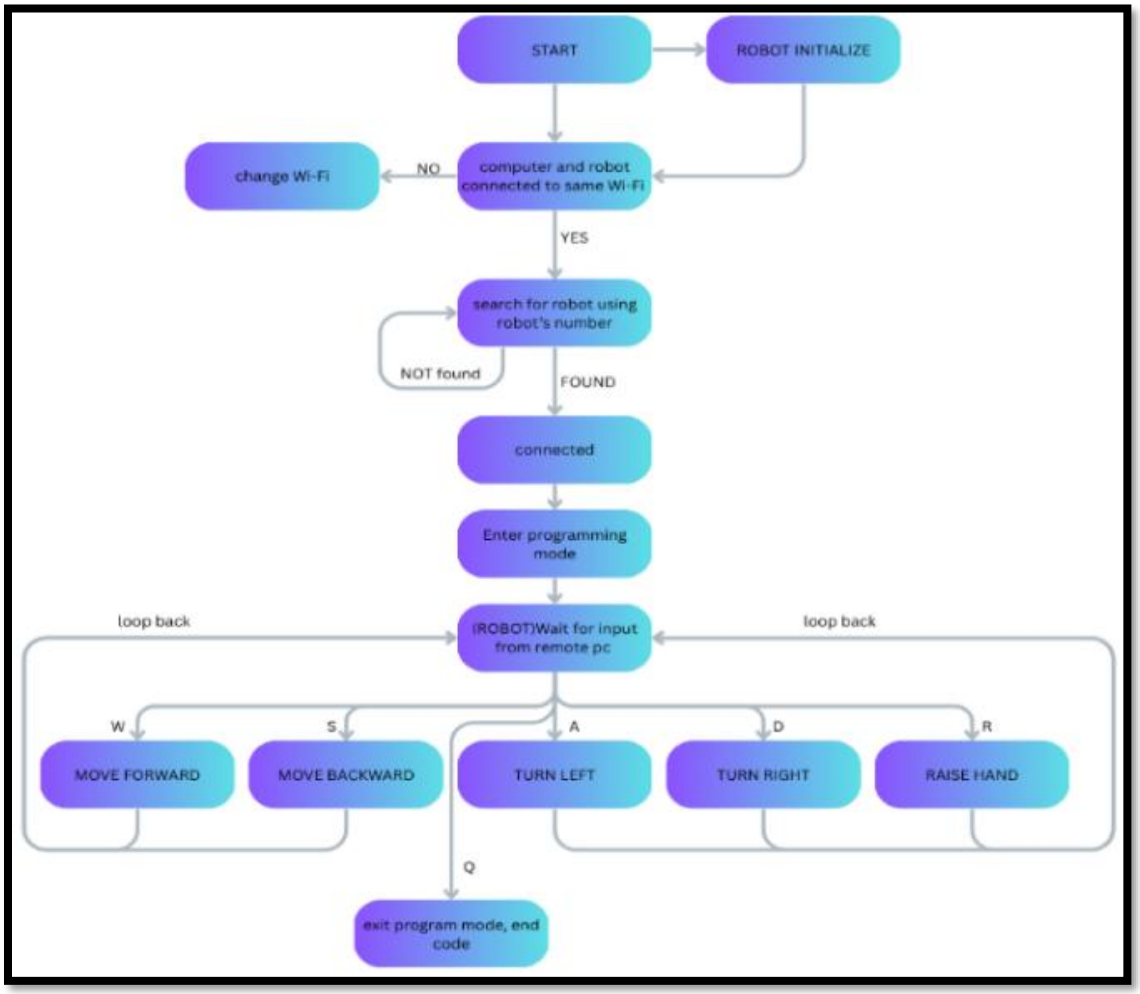
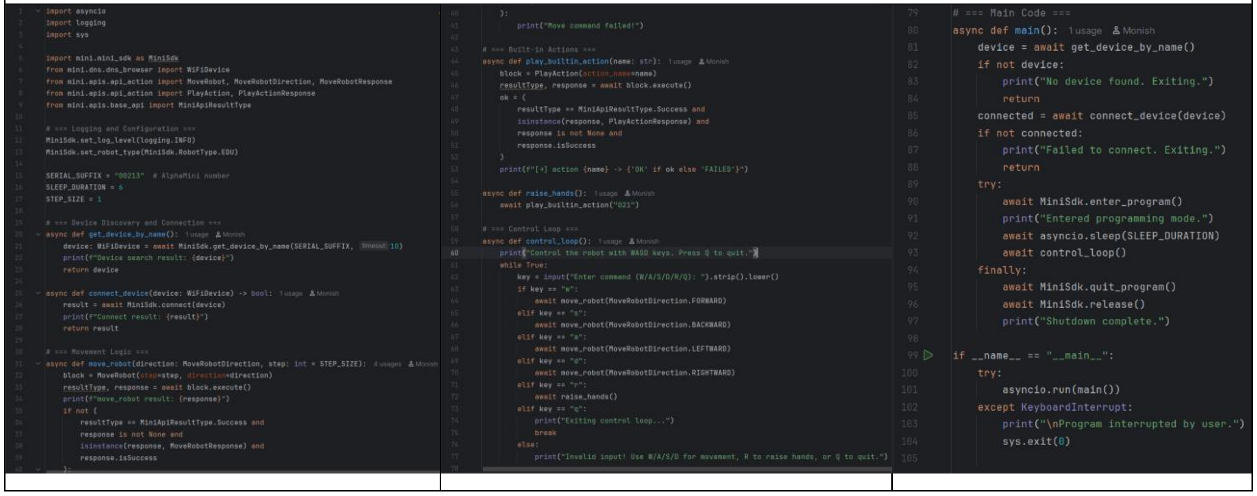


Figure 6: Manual Navigation Logic Flowchart

Code for Navigation – Manual control (based on flowchart)


```python
import asyncio
import logging
import sys

import mini.mini_sdk as MiniSdk
from mini.dns.dns_browser import WiFiDevice
from mini.apis.api_action import MoveRobot, MoveRobotDirection, MoveRobotResponse
from mini.apis.api_action import PlayAction, PlayActionResponse
from mini.apis.base_api import MiniApiResultType

# === Logging and Configuration ===
MiniSdk.set_log_level(logging.INFO)
MiniSdk.set_robot_type(MiniSdk.RobotType.EDU)

SERIAL_SUFFIX = "00213"  # AlphaMini number
SLEEP_DURATION = 6
STEP_SIZE = 1

# === Device Discovery and Connection ===
async def get_device_by_name():
    device: WiFiDevice = await MiniSdk.get_device_by_name(SERIAL_SUFFIX, 10)
    print(f"Device search result: {device}")
    return device

async def connect_device(device: WiFiDevice) -> bool:
    result = await MiniSdk.connect(device)
    print(f"Connect result: {result}")
    return result

# === Movement Logic ===
async def move_robot(direction: MoveRobotDirection, step: int = STEP_SIZE):
    block = MoveRobot(step=step, direction=direction)
    resultType, response = await block.execute()
    print(f"move_robot result: {response}")
    if not (
        resultType == MiniApiResultType.Success and
        response is not None and
        isinstance(response, MoveRobotResponse) and
        response.isSuccess
    ):
        print("Move command failed!")

# === Built-in Actions ===
async def play_builtin_action(name: str):
    block = PlayAction(action_name=name)
    resultType, response = await block.execute()
    ok = (
        resultType == MiniApiResultType.Success and
        isinstance(response, PlayActionResponse) and
        response is not None and
        response.isSuccess
    )
    print(f"[+] action {name} -> {'OK' if ok else 'FAILED'}")

async def raise_hands():
    await play_builtin_action("021")

# === Control Loop ===
async def control_loop():
    print("Control the robot with WASD keys. Press Q to quit.")
    while True:
        key = input("Enter command (W/A/S/D/R/Q): ").strip().lower()
        if key == "w":
            await move_robot(MoveRobotDirection.FORWARD)
        elif key == "s":
            await move_robot(MoveRobotDirection.BACKWARD)
        elif key == "a":
            await move_robot(MoveRobotDirection.LEFTWARD)
        elif key == "d":
            await move_robot(MoveRobotDirection.RIGHTWARD)
        elif key == "r":
            await raise_hands()
        elif key == "q":
            print("Exiting control loop...")
            break
        else:
            print("Invalid input! Use W/A/S/D for movement, R to raise hands, or Q to quit.")

# === Main Code ===
async def main():
    device = await get_device_by_name()
    if not device:
        print("No device found. Exiting.")
        return
    connected = await connect_device(device)
    if not connected:
        print("Failed to connect. Exiting.")
        return
    try:
        await MiniSdk.enter_program()
        print("Entered programming mode.")
        await asyncio.sleep(SLEEP_DURATION)
        await control_loop()
    finally:
        await MiniSdk.quit_program()
        await MiniSdk.release()
        print("Shutdown complete.")

if __name__ == "__main__":
    try:
        asyncio.run(main())
    except KeyboardInterrupt:
        print("\nProgram interrupted by user.")
        sys.exit(0)
```

Table 6: Code for Navigation - Manual control

III) Reflection

The behaviors or features that were made before as a standalone were great and ideal, but when trying to improve them as originally intended, major problems started to arise. The major problem being the accessibility of Alpha Mini feature given to students, I stumbled across this while trying to get the data of picture from mini to my remote computer (this was done at the same time navigation was done). The AlphaMini given to students was stripped of its *mini.apis.api_media* and *mini.apis.api_vision* modules. These modules are responsible for transfer of images or video streaming through the SDK socket. The retrieval of pictures taken can be done from the AlphaMini mobile app, it is more counterproductive to the idea which focuses on reducing work to take attendance. Assuming the reason for having an Edu version of Alpha Mini and not a DEV version is data safety and to not give anyone a literal walking camera and stream it to a remote computer. As the concept is to have a DEV version, I will proceed with the code for dev version and explain that in upcoming courseworks.

Other problems that arose during NAVIGATION are; connectivity issues and I found sometimes the AlphaMini loses balance by itself, therefore some form of safety measures are also required.

The code required more fine tuning and isn't at its finalized version so it's good for now, after combining other code the code should look cleaner and logical.

Student#2 – La Min Aung

In this coursework 3, I have decided to do those automatic navigation movements. And also, I have tried for autonomous navigation plus obstacle avoidance of AlphaMini. So, in coming coursework my aim is to do the speech recognition as much as I can.

I) Navigation without sensing

For this section, I focused on coding the AlphaMini to be capable of autonous navigation. Adopting and developing further on the manual navigation code, some features were added.

Since AlphaMini can't move instanteously, I have put the sleep duration which means make a pause between actions just for safety. I set the timer for 2 seconds stop after each actions eg, move forward stop 2 seconds and turn right stop 2 seconds, turn left and then stop for 2seconds again. And also I have changed the step size to 2 from 1 because it look more clear that robot move forward or turning around.

This template of autonomous navigation aims to be a building block for the final presentation **path planning.** The idea is to lay a foundational code that can be edited and extended for any automated movements that will align with the final presentation's storyboard.

The flowchart below shows the logic and workflow of the autonomous navigation –

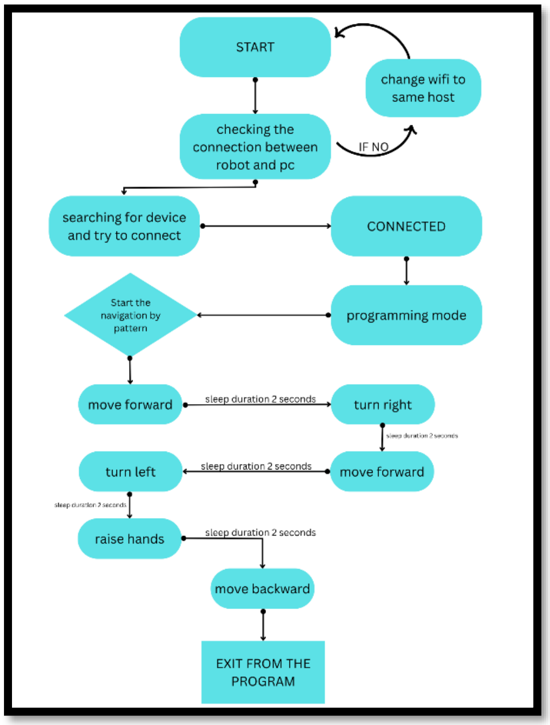
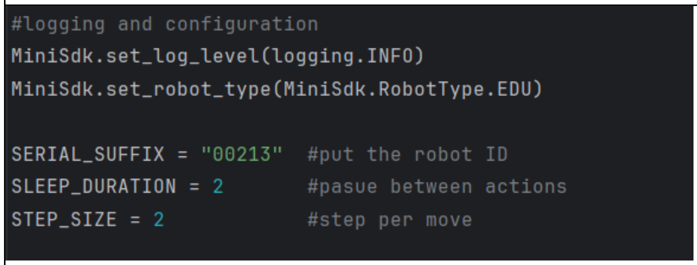


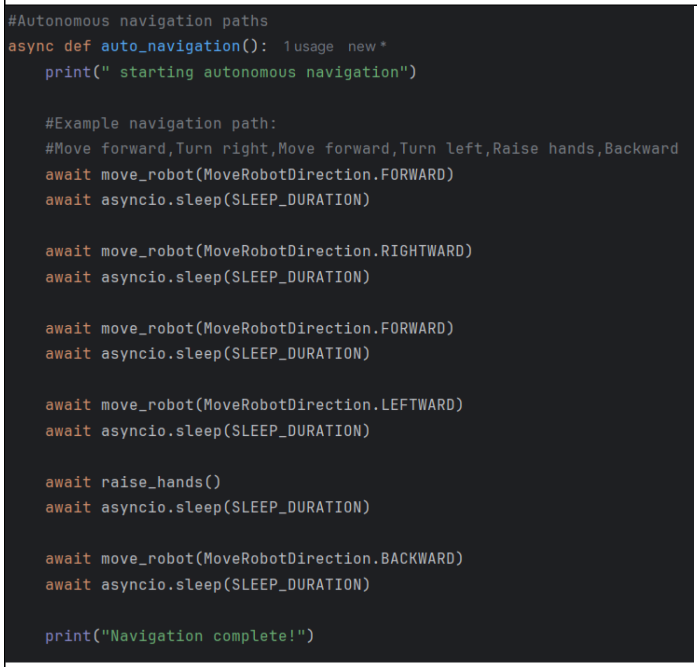
Figure 7: Autonomous Navigation Logic Flowchart

Code for Navigation – Automation (based on flowchart)



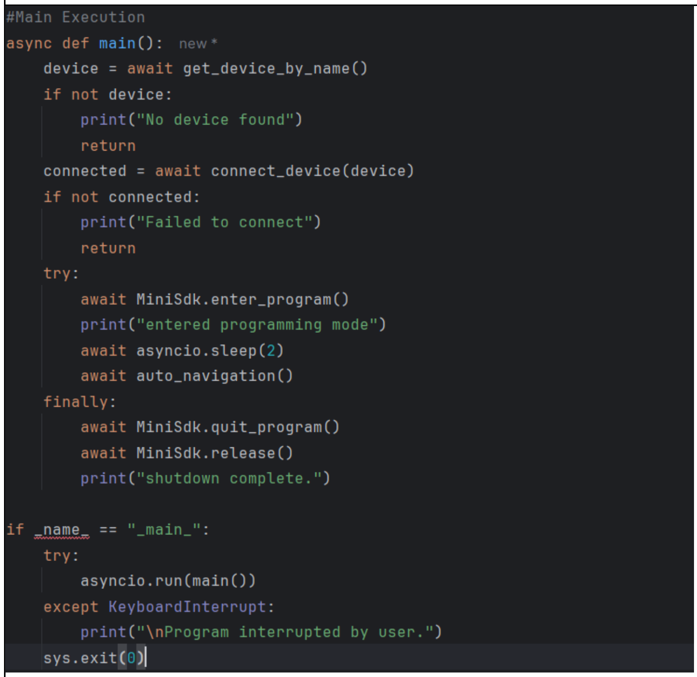
```
#logging and configuration
MiniSdk.set_log_level(logging.INFO)
MiniSdk.set_robot_type(MiniSdk.RobotType.EDU)

SERIAL_SUFFIX = "00213"  #put the robot ID
SLEEP_DURATION = 2       #pasue between actions
STEP_SIZE = 2            #step per move
```

After testing and calibration, I set sleep to 2 seconds for obvious and visible movements of the AlphaMini in real time



```
#Autonomous navigation paths
async def auto_navigation():  1 usage  new *
    print(" starting autonomous navigation")

    #Example navigation path:
    #Move forward,Turn right,Move forward,Turn left,Raise hands,Backward
    await move_robot(MoveRobotDirection.FORWARD)
    await asyncio.sleep(SLEEP_DURATION)

    await move_robot(MoveRobotDirection.RIGHTWARD)
    await asyncio.sleep(SLEEP_DURATION)

    await move_robot(MoveRobotDirection.FORWARD)
    await asyncio.sleep(SLEEP_DURATION)

    await move_robot(MoveRobotDirection.LEFTWARD)
    await asyncio.sleep(SLEEP_DURATION)

    await raise_hands()
    await asyncio.sleep(SLEEP_DURATION)

    await move_robot(MoveRobotDirection.BACKWARD)
    await asyncio.sleep(SLEEP_DURATION)

    print("Navigation complete!")
```

In this custom paths way navigation code, “asyncio” is used to ensure that the movement pattern is executed carefully and sequentially. The “await” calls, similar to “asyncio.sleep”, ensure that the progran waits not blocking for each AlphaMini’s action to complete before initiating the next move. Thus keeping control thread as an entire process from freezing.



```
#Main Execution
async def main():  new *
    device = await get_device_by_name()
    if not device:
        print("No device found")
        return
    connected = await connect_device(device)
    if not connected:
        print("Failed to connect")
        return
    try:
        await MiniSdk.enter_program()
        print("entered programming mode")
        await asyncio.sleep(2)
        await auto_navigation()
    finally:
        await MiniSdk.quit_program()
        await MiniSdk.release()
        print("shutdown complete.")

if __name__ == "__main__":
    try:
        asyncio.run(main())
    except KeyboardInterrupt:
        print("\nProgram interrupted by user.")
    sys.exit(0)
```

Main loop for execution of autonomous navigation

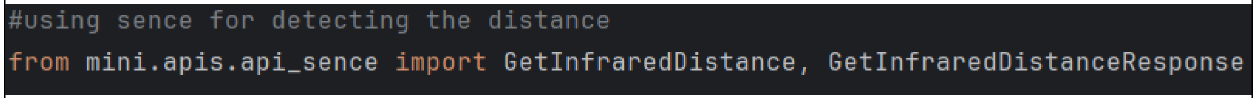
Table 7: Code for Navigation - Automation

II) Navigation – Obstacle Avoidance

This next section explains how I implemented the obstacle avoidance logic for AlphaMini. Using AlphaMini's infrared distance sensors and movements APIs.

<u>Code for navigation – obstacle avoidance with Code logic explanation</u>

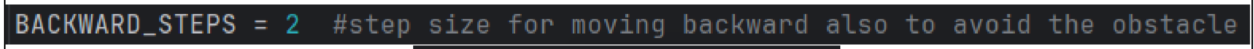
Obstacle Detection:



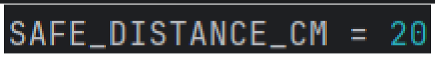
```
#using sence for detecting the distance
from mini.apis.api_sence import GetInfraredDistance, GetInfraredDistanceResponse
```

I integrated the GID (getinfrareddistance) api to continuously show the path ahead for AlphaMini

Avoidance Strategy:



```
BACKWARD_STEPS = 2  #step size for moving backward also to avoid the obstacle
```

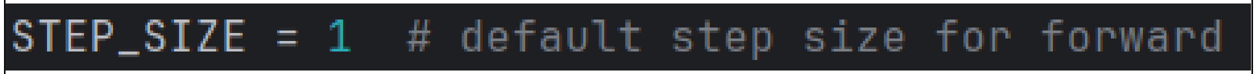


```
SAFE_DISTANCE_CM = 20
```

I have made a specific avoidance mechanism for AlphaMini which will also be efficient for narrow spaces. Upon detecting an obstacle closer than 20cm, AlphaMini will move backward to fix the distance between AlphaMini and obstacle (BACKWARD_STEPS=2) and then make a right turn to change direction away from obstacle and try to find a clear path. This ensures that before trying to avoid the immediate threat, it will retreat from it.

In summary if an obstacle is detected, the sequence is Move Backward -> Turn right -> Forward thus then resuming the goal of continuous autonomous navigation

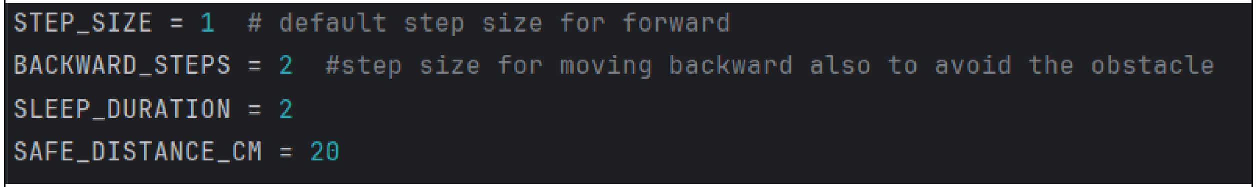
Continuous Movement:



```
STEP_SIZE = 1  # default step size for forward
```

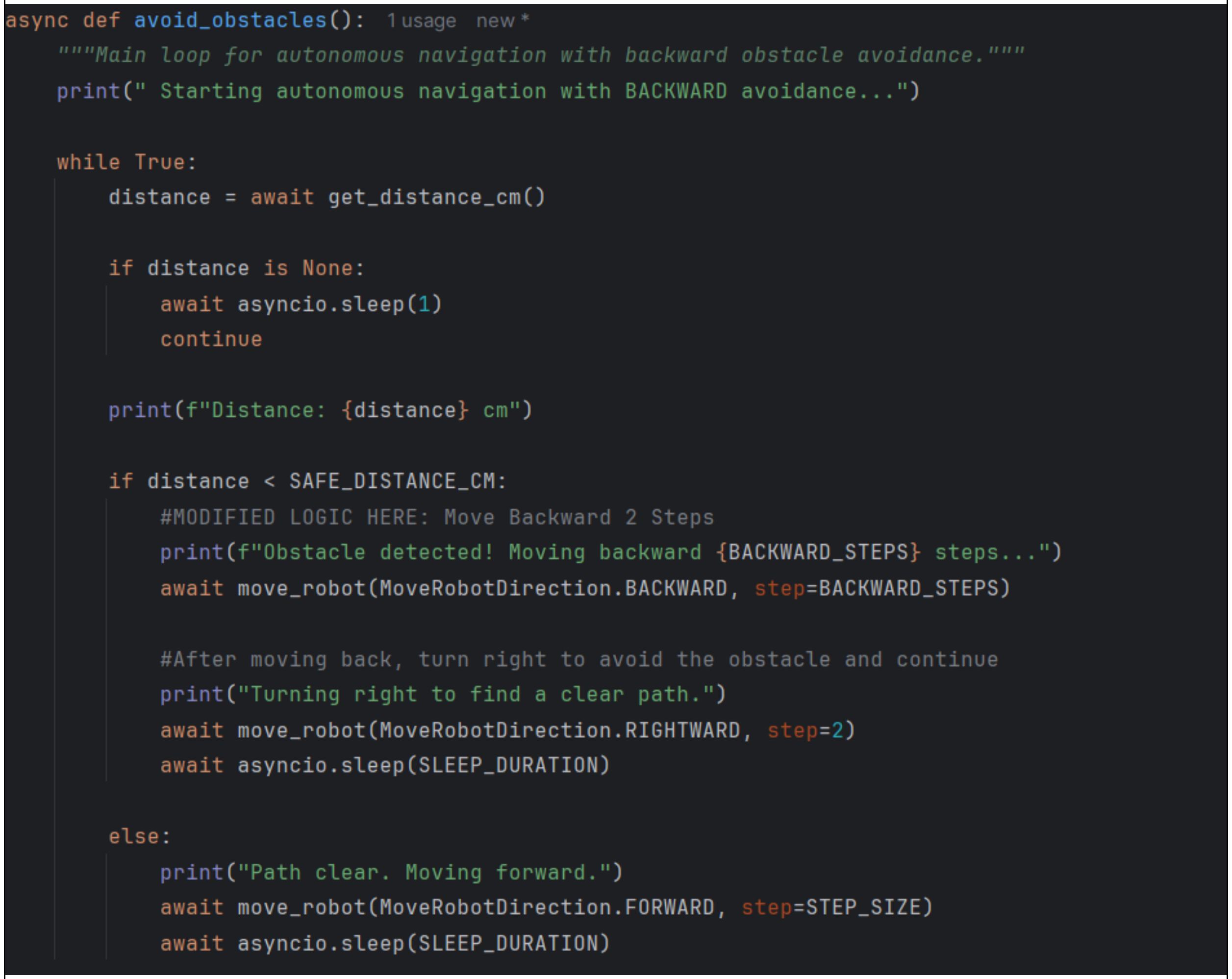
For continuous navigation I have created the main loop to keep AlphaMini moving forward by default mechanism whenever the path is clear, thus ensuring continuous exploration

Parameter configurations recap:



```
STEP_SIZE = 1  # default step size for forward
BACKWARD_STEPS = 2  #step size for moving backward also to avoid the obstacle
SLEEP_DURATION = 2
SAFE_DISTANCE_CM = 20
```

I established adjustment configuration parameters in easy way to fine-tune AlphaMini's behavior for different environments.

Main Loop:

The **while True** loop ensures that as soon as one action is complete, AlphaMini will immediately returns to detecting the distance again, which is necessary for obstacle avoidance, guaranteeing continuous obstacle avoidance operation.

```python
async def avoid_obstacles():  1 usage  new *
    """Main loop for autonomous navigation with backward obstacle avoidance."""
    print(" Starting autonomous navigation with BACKWARD avoidance...")

    while True:
        distance = await get_distance_cm()

        if distance is None:
            await asyncio.sleep(1)
            continue

        print(f"Distance: {distance} cm")

        if distance < SAFE_DISTANCE_CM:
            #MODIFIED LOGIC HERE: Move Backward 2 Steps
            print(f"Obstacle detected! Moving backward {BACKWARD_STEPS} steps...")
            await move_robot(MoveRobotDirection.BACKWARD, step=BACKWARD_STEPS)

            #After moving back, turn right to avoid the obstacle and continue
            print("Turning right to find a clear path.")
            await move_robot(MoveRobotDirection.RIGHTWARD, step=2)
            await asyncio.sleep(SLEEP_DURATION)

        else:
            print("Path clear. Moving forward.")
            await move_robot(MoveRobotDirection.FORWARD, step=STEP_SIZE)
            await asyncio.sleep(SLEEP_DURATION)
```

Table 8: Code for Navigation – Obstacle Avoidance

Obstacle avoidance logic flowchart:

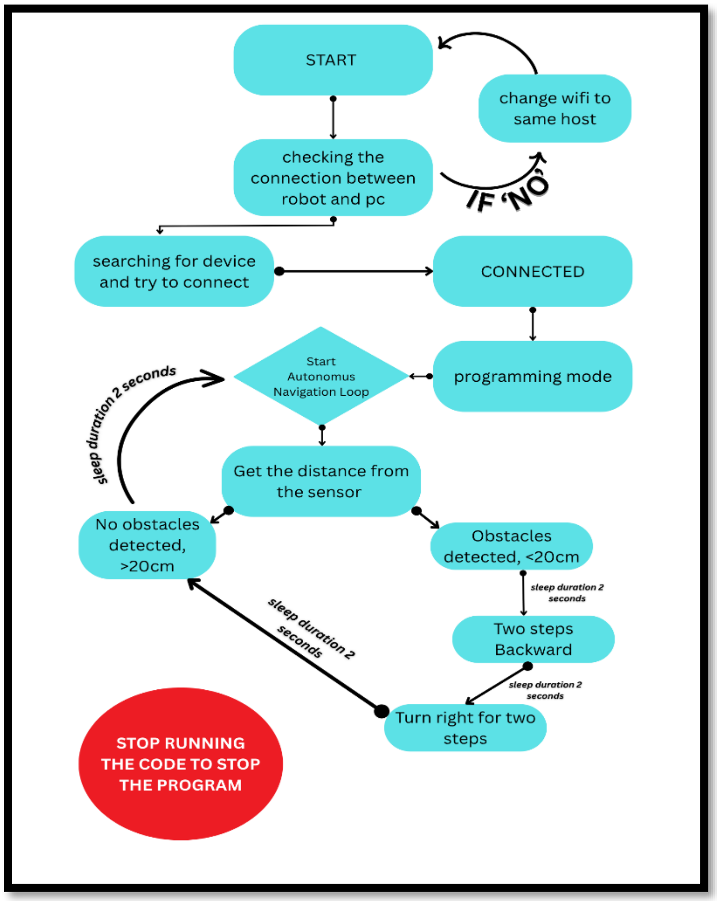

START
change wifi to same host
checking the connection between robot and pc
IF 'NO'
searching for device and try to connect
CONNECTED
programming mode
Start Autonomus Navigation Loop
sleep duration 2 seconds
Get the distance from the sensor
No obstacles detected, >20cm
Obstacles detected, <20cm
sleep duration 2 seconds
Two steps Backward
sleep duration 2 seconds
sleep duration 2 seconds
Turn right for two steps
STOP RUNNING THE CODE TO STOP THE PROGRAM

Figure 8: Obstacle Avoidance Flowchart

III) Reflection

This coursework 3 team project was primarily a study in implementation complexity and management, creation of two separate algorithms: A custom pathway autonomous navigation system and a reactive model including obstacle avoidance logic. The core lesson wasn't just about coding itself, but keep changing code and test the code multiple times until it worked. What I thought would be a quick, but linear build quickly turned into a bumpy. For the second one, I got a lot of challenging for the sensor. First, I didn't know that our AlphaMini's is the old version and also I forgot that we have to use the old version of the IDE. So, even after downloading the package for AlphaMini, the specific sensor file are not include. I have to base on the sensor data for detecting the distance. After I figure it out, my code is start working. For now I have wrote about the navigation and obstacle avoidance part.

Student#3 – Cornelius Fulvian (safety)

In this classwork 3, i have decided to do the safety features for the AlphaMini navigation. I give 3 safety features for the AlphaMini and that is Emergency Stop System, Sensor Sanity & Debounce, and Command Rate Limit & Step Bounds.

I) Navigation – Safety Feature

a. Emergency Stop System

This feature lets you instantly stop the robots movement if somethings went wrong. For example, if the AlphaMini is about to fall of the desk, you can manually stop the AlphaMini.

How it works:

1. You can press e and then enter in the terminal.
2. The code runs a background task called *estop_listener().*
3. When you type e, it sets a global flag (*estop_event*) which tells the AlphaMini to:
    - Stop all new movements.
    - Exit the main loop safely.
    - Release the motors and quit the SDK cleanly.

This feature is needed because we need to prevent any damages or accidents if the AlphaMini behaves unexpectedly.

b. Sensor Sanity and Debounce

This feature keeps the AlphaMini from reacting falsely or noisy distance readings and stop if the sensors fails repeatedly.

How it works:

1. Every time the AlphaMini reads the infrared distance:
    - The reading is checked — it must be within valid range (e.g., 1 – 5000 cm).
    - It’s stored in a small history buffer (3 recent values).
    - The median (middle value) is used to filter out noise or spikes.
2. The AlphaMini only reacts (e.g., backs up) if it sees 2 consecutive close readings (debounce).

3. If the sensor fails more than 5 times in a row, the AlphaMini safely stops (watchdog).

This feature is needed this safety avoids false triggers caused by single bad readings and keeps navigation smooth.

c. Command Rate Limit and Step Bounds

This feature prevents the AlphaMini from sending too many commands too fast or moving too far at once because this can cause unstable behavior or overload the SDK.

How it works:

1. All motion commands go through *safe_move_robot()* instead of directly using *MoveRobot*.
2. The system enforces:
    - Step clamping: Any move is limited between 1 – 6 steps (no accidental large jumps)
    - Rate limit: At least 0.4 seconds between move commands
    - E-Stop awareness: If the emergency flag is active, movement is ignored

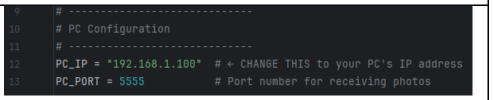
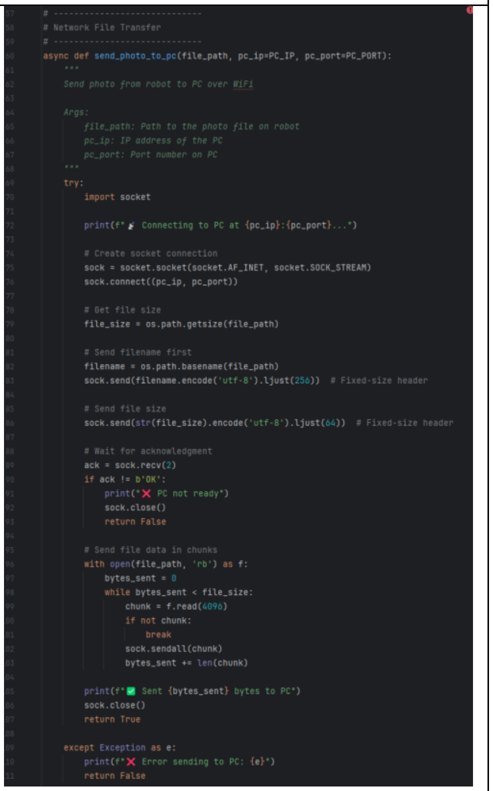
This feature matters because it keeps all motion predictable and safe

Student#4 – Joseph Santhosh Aaron

I) Compact Attendance Taking System

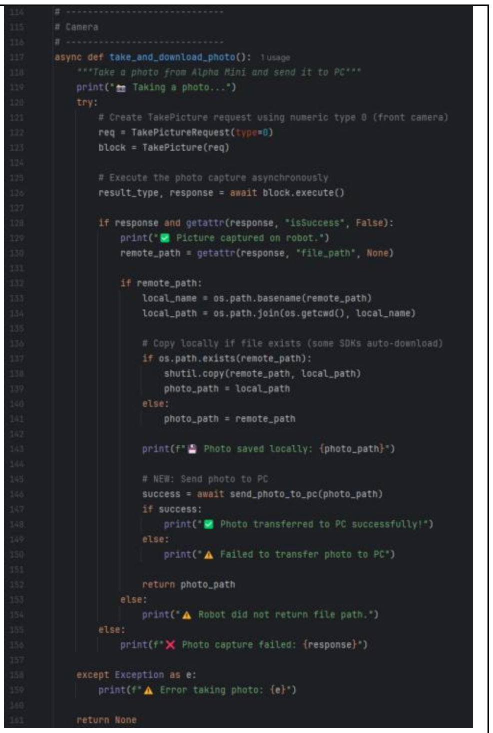
Since the Attendance Taking System was still incomplete, I was tasked to bring it to completion and test its usability. Given that the Rcat is already capable of correctly identifying QR codes and then sending it to the Remote PC, I now needed to 1. Create a script/function to decipher the qr code in legible date 2. Research and develop and approach to log that legible data for the teaching staff to later use.

After researching and deliberating on Online or Offline databases (such as SQLite3, pyzbar and even simple .csv), I came to the conclusion that excel as a logging base both is conventional as it is widely available to schools and flexible as it can also be made available online.

<u>Code for Attendance Taking System – AlphaMini side</u>	
First, I needed to make some changes to the AlphaMini side of the QR code (done in CW2) as it wasn't capable of storing images into the Remote PC via WiFi.	
Code Ideation and Substantiation	**Code with comments**
Links the AlphaMini to the PC. The PC's ip can be found via ipconfig in windows command prompt terminal	
The function **send_photos_to_pc()** formats then sends the file.	

```python
# ----------------------------
# PC Configuration
# ----------------------------
PC_IP = "192.168.1.100"  # ← CHANGE THIS to your PC's IP address
PC_PORT = 5555           # Port number for receiving photos
```

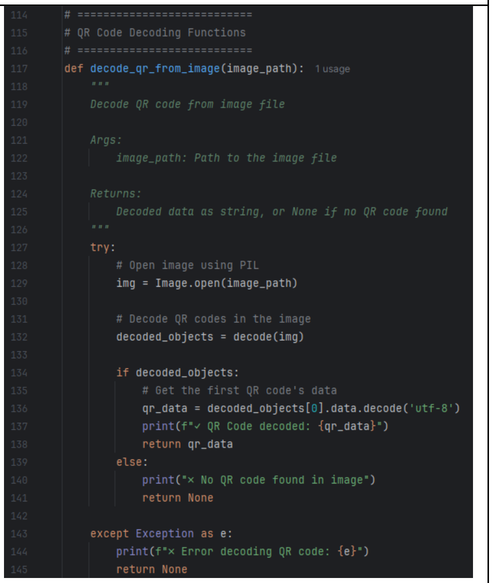
```python
# ----------------------------
# Network File Transfer
# ----------------------------
async def send_photo_to_pc(file_path, pc_ip=PC_IP, pc_port=PC_PORT):
    """
    Send photo from robot to PC over WiFi

    Args:
        file_path: Path to the photo file on robot
        pc_ip: IP address of the PC
        pc_port: Port number on PC
    """
    try:
        import socket

        print(f"📡 Connecting to PC at {pc_ip}:{pc_port}...")

        # Create socket connection
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((pc_ip, pc_port))

        # Get file size
        file_size = os.path.getsize(file_path)

        # Send filename first
        filename = os.path.basename(file_path)
        sock.send(filename.encode('utf-8').ljust(256))  # Fixed-size header

        # Send file size
        sock.send(str(file_size).encode('utf-8').ljust(64))  # Fixed-size header

        # Wait for acknowledgment
        ack = sock.recv(2)
        if ack != b'OK':
            print("❌ PC not ready")
            sock.close()
            return False

        # Send file data in chunks
        with open(file_path, 'rb') as f:
            bytes_sent = 0
            while bytes_sent < file_size:
                chunk = f.read(4096)
                if not chunk:
                    break
                sock.sendall(chunk)
                bytes_sent += len(chunk)

        print(f"✅ Sent {bytes_sent} bytes to PC")
        sock.close()
        return True

    except Exception as e:
        print(f"❌ Error sending to PC: {e}")
        return False
```

Then I modfied the existing **take_and_download_photo()** function to send the photo to the PC via WiFi.	

```python
# ----------------------------
# Camera
# ----------------------------
async def take_and_download_photo():  1 usage
    """Take a photo from Alpha Mini and send it to PC"""
    print("📷 Taking a photo...")
    try:
        # Create TakePicture request using numeric type 0 (front camera)
        req = TakePictureRequest(type=0)
        block = TakePicture(req)

        # Execute the photo capture asynchronously
        result_type, response = await block.execute()

        if response and getattr(response, "isSuccess", False):
            print("✅ Picture captured on robot.")
            remote_path = getattr(response, "file_path", None)

            if remote_path:
                local_name = os.path.basename(remote_path)
                local_path = os.path.join(os.getcwd(), local_name)

                # Copy locally if file exists (some SDKs auto-download)
                if os.path.exists(remote_path):
                    shutil.copy(remote_path, local_path)
                    photo_path = local_path
                else:
                    photo_path = remote_path

                print(f"💾 Photo saved locally: {photo_path}")

                # NEW: Send photo to PC
                success = await send_photo_to_pc(photo_path)
                if success:
                    print("✅ Photo transferred to PC successfully!")
                else:
                    print("⚠ Failed to transfer photo to PC")

                return photo_path
            else:
                print("⚠ Robot did not return file path.")
        else:
            print(f"❌ Photo capture failed: {response}")

    except Exception as e:
        print(f"⚠ Error taking photo: {e}")

    return None
```

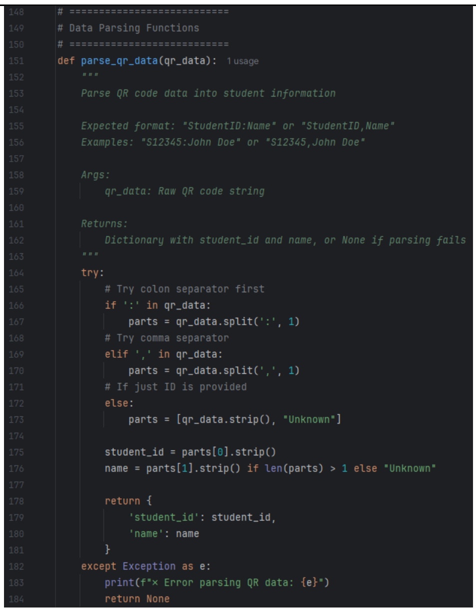
Table 9: Code for Attendance Taking System – AlphaMini side

After syncing the AlphaMini to the PC and testing if images were sucessfully uploaded. The next line of order was to code the external script that would handle the deciphering and logging part.

Code for Attendance Taking System – Remote PC side	
Code Ideation and Substantiation	**Code with comments**
First, I scripted the code logic to decode the Qr code from and a stored image in a location on the remote PC and named it **decode_qr_from_image()**	

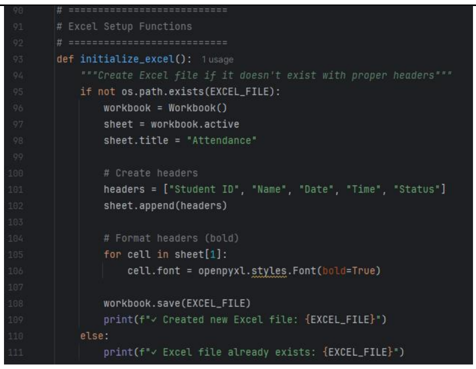
```python
# ===========================
# QR Code Decoding Functions
# ===========================
def decode_qr_from_image(image_path):  1 usage
    """
    Decode QR code from image file

    Args:
        image_path: Path to the image file

    Returns:
        Decoded data as string, or None if no QR code found
    """
    try:
        # Open image using PIL
        img = Image.open(image_path)

        # Decode QR codes in the image
        decoded_objects = decode(img)

        if decoded_objects:
            # Get the first QR code's data
            qr_data = decoded_objects[0].data.decode('utf-8')
            print(f"✓ QR Code decoded: {qr_data}")
            return qr_data
        else:
            print("× No QR code found in image")
            return None

    except Exception as e:
        print(f"× Error decoding QR code: {e}")
        return None
```

Next, I scripted a function **parse_qr_data()** to format the data so that it is prepared to be logged into my excel sheet according to the designated headers.



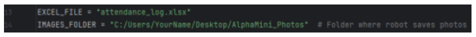
```python
# ===========================
# Data Parsing Functions
# ===========================
def parse_qr_data(qr_data):  1 usage
    """
    Parse QR code data into student information

    Expected format: "StudentID:Name" or "StudentID,Name"
    Examples: "S12345:John Doe" or "S12345,John Doe"

    Args:
        qr_data: Raw QR code string

    Returns:
        Dictionary with student_id and name, or None if parsing fails
    """
    try:
        # Try colon separator first
        if ':' in qr_data:
            parts = qr_data.split(':', 1)
        # Try comma separator
        elif ',' in qr_data:
            parts = qr_data.split(',', 1)
        # If just ID is provided
        else:
            parts = [qr_data.strip(), "Unknown"]

        student_id = parts[0].strip()
        name = parts[1].strip() if len(parts) > 1 else "Unknown"

        return {
            'student_id': student_id,
            'name': name
        }
    except Exception as e:
        print(f"× Error parsing QR data: {e}")
        return None
```

Following which I created a function **initialize_excel()** for the script to access an excel file and a means to create an excel file if the excel file for example “Class_A01.xlsx” did not exist or got corrupted. This ensures that no matter what the students’ attendance is taken even if it is another file (for staff to figure out).



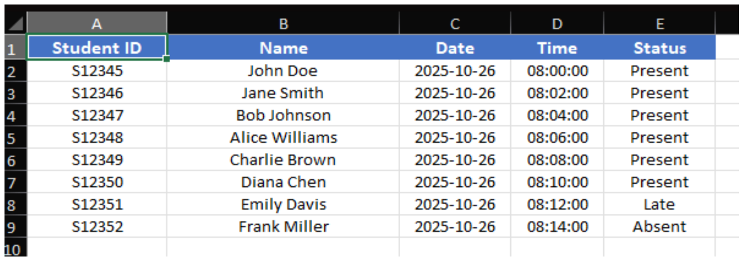
```python
# ===========================
# Excel Setup Functions
# ===========================
def initialize_excel():  1 usage
    """Create Excel file if it doesn't exist with proper headers"""
    if not os.path.exists(EXCEL_FILE):
        workbook = Workbook()
        sheet = workbook.active
        sheet.title = "Attendance"

        # Create headers
        headers = ["Student ID", "Name", "Date", "Time", "Status"]
        sheet.append(headers)

        # Format headers (bold)
        for cell in sheet[1]:
            cell.font = openpyxl.styles.Font(bold=True)

        workbook.save(EXCEL_FILE)
        print(f"✓ Created new Excel file: {EXCEL_FILE}")
    else:
        print(f"✓ Excel file already exists: {EXCEL_FILE}")
```

For user friendliness, “attendance_log.xlsx” line is the only line needed to be changed for the whole script to call the specific file. eg: changing to“Class_A02.xlsx”



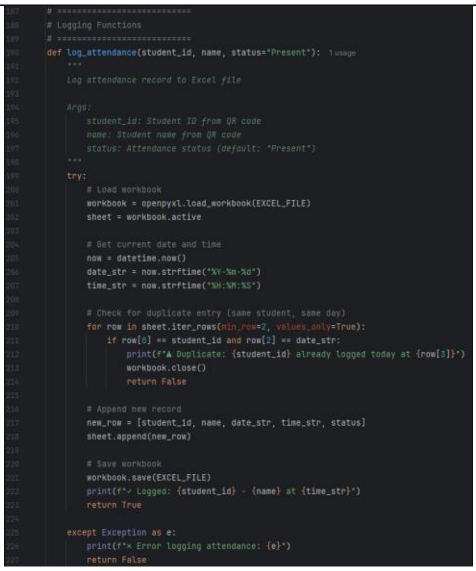
```python
EXCEL_FILE = "attendance_log.xlsx"
IMAGES_FOLDER = "C:/Users/YourName/Desktop/AlphaMini_Photos"  # Folder where robot saves photos
```

Now that the data is prepped, the function **log_attendance()** takes in relevant variables and then stores it in the excel sheet and it should look something like the image attached below in figure.

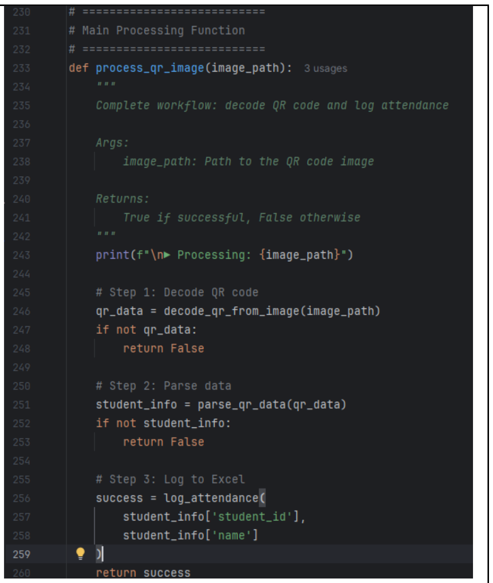
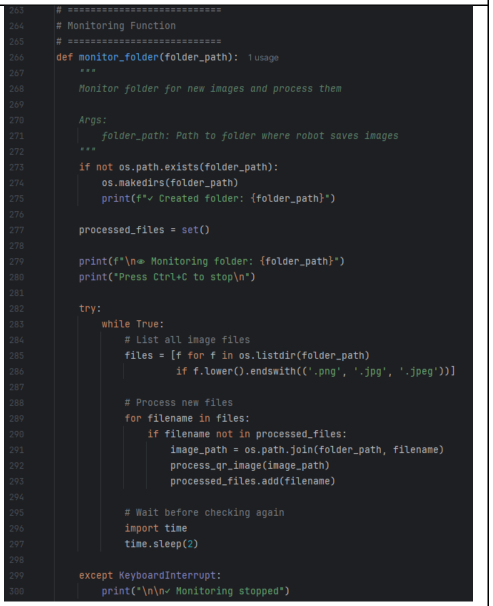


	A	B	C	D	E
1	Student ID	Name	Date	Time	Status
2	S12345	John Doe	2025-10-26	08:00:00	Present
3	S12346	Jane Smith	2025-10-26	08:02:00	Present
4	S12347	Bob Johnson	2025-10-26	08:04:00	Present
5	S12348	Alice Williams	2025-10-26	08:06:00	Present
6	S12349	Charlie Brown	2025-10-26	08:08:00	Present
7	S12350	Diana Chen	2025-10-26	08:10:00	Present
8	S12351	Emily Davis	2025-10-26	08:12:00	Late
9	S12352	Frank Miller	2025-10-26	08:14:00	Absent
10					

Figure 9: Excel Sample

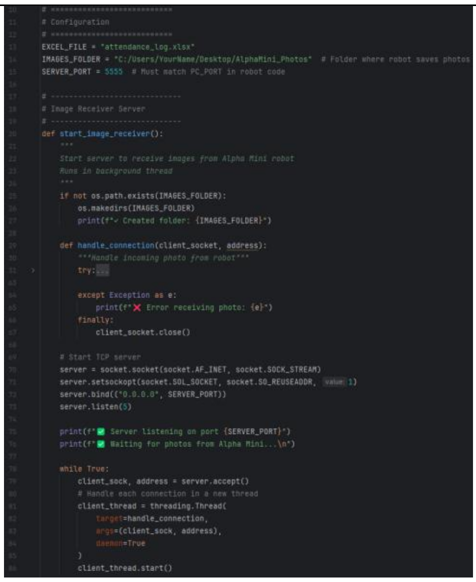
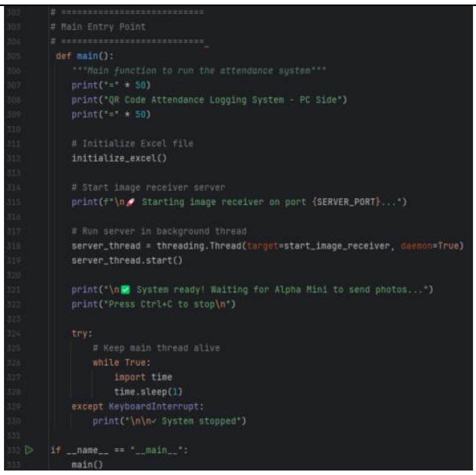
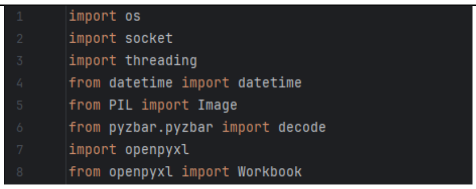


```python
# ===========================
# Logging Functions
# ===========================
def log_attendance(student_id, name, status="Present"):  1 usage
    """
    Log attendance record to Excel file

    Args:
        student_id: Student ID from QR code
        name: Student name from QR code
        status: Attendance status (default: "Present")
    """
    try:
        # Load workbook
        workbook = openpyxl.load_workbook(EXCEL_FILE)
        sheet = workbook.active

        # Get current date and time
        now = datetime.now()
        date_str = now.strftime("%Y-%m-%d")
        time_str = now.strftime("%H:%M:%S")

        # Check for duplicate entry (same student, same day)
        for row in sheet.iter_rows(min_row=2, values_only=True):
            if row[0] == student_id and row[2] == date_str:
                print(f"⚠ Duplicate: {student_id} already logged today at {row[3]}")
                workbook.close()
                return False

        # Append new record
        new_row = [student_id, name, date_str, time_str, status]
        sheet.append(new_row)

        # Save workbook
        workbook.save(EXCEL_FILE)
        print(f"✓ Logged: {student_id} - {name} at {time_str}")
        return True

    except Exception as e:
        print(f"× Error logging attendance: {e}")
        return False
```

I programmed the function **process_qr_image()** to incorporate the decoding parsing and logging into one holistic function, not only for debugging but for cleaner more logical code.	
I also needed a way to check for new images of qr codes uploaded and then run the decode->parsing->logging loop again. So I created the function **monitor_folder()** that looks through the folder in the os for files that havent been processed before.	

```python
# ===========================
# Main Processing Function
# ===========================
def process_qr_image(image_path):  3 usages
    """
    Complete workflow: decode QR code and log attendance

    Args:
        image_path: Path to the QR code image

    Returns:
        True if successful, False otherwise
    """
    print(f"\n► Processing: {image_path}")

    # Step 1: Decode QR code
    qr_data = decode_qr_from_image(image_path)
    if not qr_data:
        return False

    # Step 2: Parse data
    student_info = parse_qr_data(qr_data)
    if not student_info:
        return False

    # Step 3: Log to Excel
    success = log_attendance(
        student_info['student_id'],
        student_info['name']
    )
    return success
```

```python
# ==========================
# Monitoring Function
# ==========================
def monitor_folder(folder_path):  1 usage
    """
    Monitor folder for new images and process them

    Args:
        folder_path: Path to folder where robot saves images
    """
    if not os.path.exists(folder_path):
        os.makedirs(folder_path)
        print(f"✓ Created folder: {folder_path}")

    processed_files = set()

    print(f"\n👁 Monitoring folder: {folder_path}")
    print("Press Ctrl+C to stop\n")

    try:
        while True:
            # List all image files
            files = [f for f in os.listdir(folder_path)
                         if f.lower().endswith(('.png', '.jpg', '.jpeg'))]

            # Process new files
            for filename in files:
                if filename not in processed_files:
                    image_path = os.path.join(folder_path, filename)
                    process_qr_image(image_path)
                    processed_files.add(filename)

            # Wait before checking again
            import time
            time.sleep(2)

    except KeyboardInterrupt:
        print("\n\n✓ Monitoring stopped")
```

This section ties in with the code previously in the AlphaMini which communicates with the Remote PC and then sends the image of the QR code, this is still in its early phase and is slightly janky but will be improved on and optimized when better solutions are found.	
This is just the main loop that runs all the functions together logically and runs the server for images.	
Since this script runs on pyzbar library to be capable of decoding the QR code it is essential for the PC to have it set up via "**pip install pyzbar pillow openpyxl**"	

```python
# ============================
# Configuration
# ============================
EXCEL_FILE = "attendance_log.xlsx"
IMAGES_FOLDER = "C:/Users/YourName/Desktop/AlphaMini_Photos"  # Folder where robot saves photos
SERVER_PORT = 5555  # Must match PC_PORT in robot code

# ----------------------------
# Image Receiver Server
# ----------------------------
def start_image_receiver():
    """
    Start server to receive images from Alpha Mini robot
    Runs in background thread
    """
    if not os.path.exists(IMAGES_FOLDER):
        os.makedirs(IMAGES_FOLDER)
        print(f"✓ Created folder: {IMAGES_FOLDER}")

    def handle_connection(client_socket, address):
        """Handle incoming photo from robot"""
        try:...

        except Exception as e:
            print(f"❌ Error receiving photo: {e}")
        finally:
            client_socket.close()

    # Start TCP server
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, value: 1)
    server.bind(("0.0.0.0", SERVER_PORT))
    server.listen(5)

    print(f"✅ Server listening on port {SERVER_PORT}")
    print(f"✅ Waiting for photos from Alpha Mini...\n")

    while True:
        client_sock, address = server.accept()
        # Handle each connection in a new thread
        client_thread = threading.Thread(
            target=handle_connection,
            args=(client_sock, address),
            daemon=True
        )
        client_thread.start()
```

```python
# ============================
# Main Entry Point
# ============================
def main():
    """Main function to run the attendance system"""
    print("=" * 50)
    print("QR Code Attendance Logging System - PC Side")
    print("=" * 50)

    # Initialize Excel file
    initialize_excel()

    # Start image receiver server
    print(f"\n🚀 Starting image receiver on port {SERVER_PORT}...")

    # Run server in background thread
    server_thread = threading.Thread(target=start_image_receiver, daemon=True)
    server_thread.start()

    print("\n✅ System ready! Waiting for Alpha Mini to send photos...")
    print("Press Ctrl+C to stop\n")

    try:
        # Keep main thread alive
        while True:
            import time
            time.sleep(1)
    except KeyboardInterrupt:
        print("\n\n✓ System stopped")

if __name__ == "__main__":
    main()
```

```python
import os
import socket
import threading
from datetime import datetime
from PIL import Image
from pyzbar.pyzbar import decode
import openpyxl
from openpyxl import Workbook
```

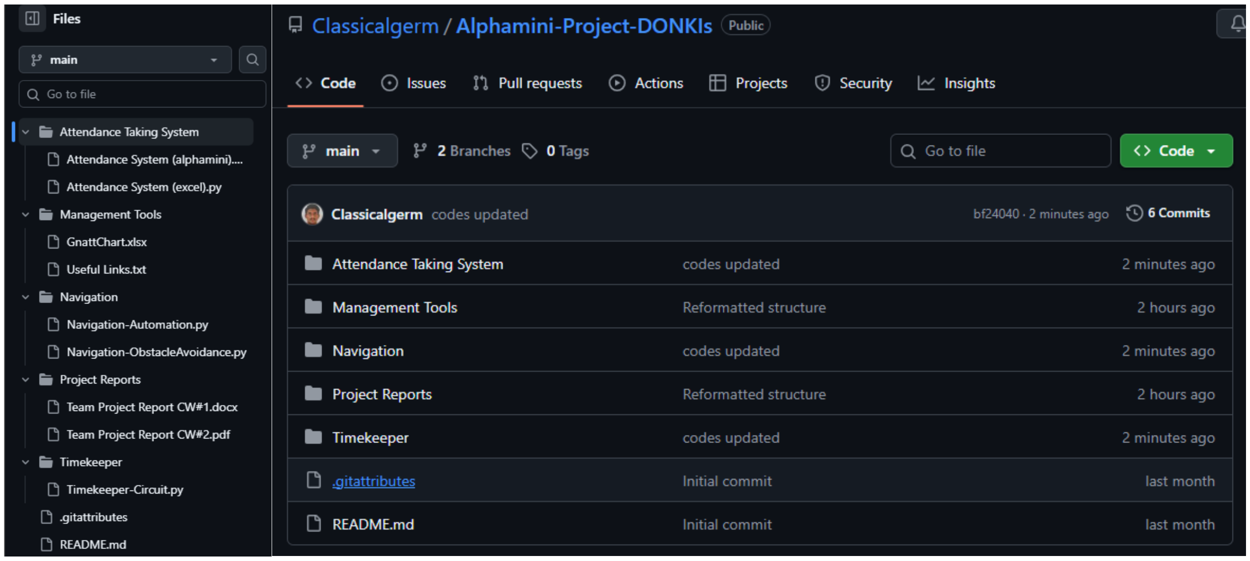
Table 10: Code for Attendance Taking System – Remote PC side

As seen previously in figure 9, the excel sheet logs attendance data in a certain format as such it is imperative that the QR code generated is in order and is labelled; Student ID: <enter ID> and Name: <enter ID>. The other data points; date, time and status is handled by the PC.

It is possible in future to create formulas in the excel sheet that once a time threshold is met the student is marked late, but that is not of utmost importance right now.

II) Navigation – Groundwork

Since my primary role is to handle the groundwork, I collated the code files for 1.Navigation- (manual control, automation, obstacle avoidance and safety), 2.Timekeeping, 3.Attendance taking system (qr scanning and data logging) into GitHub in a clear and comprehensive manner as seen below in figure 11 the


Files

main

Go to file

- Attendance Taking System
  - Attendance System (alphamini)....
  - Attendance System (excel).py
- Management Tools
  - GnattChart.xlsx
  - Useful Links.txt
- Navigation
  - Navigation-Automation.py
  - Navigation-ObstacleAvoidance.py
- Project Reports
  - Team Project Report CW#1.docx
  - Team Project Report CW#2.pdf
- Timekeeper
  - Timekeeper-Circuit.py
- .gitattributes
- README.md

Classicalgerm / Alphamini-Project-DONKIs Public

Code Issues Pull requests Actions Projects Security Insights

main 2 Branches 0 Tags Go to file Code

Classicalgerm codes updated bf24040 · 2 minutes ago 6 Commits

Attendance Taking System	codes updated	2 minutes ago
Management Tools	Reformatted structure	2 hours ago
Navigation	codes updated	2 minutes ago
Project Reports	Reformatted structure	2 hours ago
Timekeeper	codes updated	2 minutes ago
.gitattributes	Initial commit	last month
README.md	Initial commit	last month

Figure 10: GitHub Updated

There were also significant updates made to the work breakdown structure and Trello team management board.

III) Reflection

Surprisingly, getting the AlphaMini to communicate with the Remote PC was fairly easy but the QR images captured tend to have a lot of blur due to motion, lens glare (due to the qr code being displayed on my phone's LCD) and artifacting (no clue could be processing errors) and the sample sizes varied. Considering the lens glare as a serious issue, a workaround for real life application will need to be found.

It was in my plans to actually design the IDLE Hub in Fusion360 so that a major portion of the 3D Printing can be dealt with, however due to time constraints progress on that front was minimal.

All in all, since the group was primarily focused on the Navigation aspect of the AlphaMini, time with the AlphaMini to work on other features was limited, but glad that the features developed work despite some hiccups.

2.2 Remaining Work in terms of Navigation

Identification of any remaining work **related to the navigation aspect.** Provide details how the different components are/will be integrated (overview). Mention about any risks involved and how those risks can be mitigated (overview).

For the final project presentation, we have yet to storyboard and design the test simulation map (mock scaled classroom), thus in terms of navigation we have yet to start on mapping and path planning, but this is definitely on our agenda of tasks to be worked on.

For navigation specifically some movement ideas we have for our presentation storyboard is:

- Startup: something small to showcase automation
- Start of class: from idle hub to door of classroom for Attendance taking showcase
- Mid class: a pre-defined path to a table from idle hub for Language assistant feature showcase
- End class: on a pre-defined path back to idle hub we can place obstacles to showcase obstacle avoidance
- End class: need some way to showcase low battery safety feature (yet to decide)

It is also possible that we may tweak certain aspects of manual navigation to better suit the final presentation eg; pressing shift+forward on keyboard allows the Rcat to move forward indefinitely and the idea of integrating some navigational features to be handled from the Alpha Mini app itself, which will be more user friendly considering the end user are school teachers.

As of now we are programming and developing features standalone and are yet to test and see how when put together, they may interact and collide with each other. This is a risk we are running as continuous testing while development often mean troubleshooting and debugging is not as daunting a task as it is easier to pinpoint reasons failure.

3. Conclusion

Considering the work breakdown structure as a guide, we have 1. Made significant progress and only lack context specific functionalities for navigation (1 of the 2 core function of Rcat). 2. Achieved full timekeeping and attendance taking (2 of 4 feature functions of Rcat) we seem to be on track and covered the lag from earlier.

4. All team member signatures

Name	Signature
Monish Naresh Kumar	[illegible]
La Min Aung	[illegible]
Cornelius Fulvian	[illegible]
Joseph Santhosh Aaron	[illegible]

Table 11: Members' Signatures

References

NIE, NTU . (n.d.). From libguides : https://libguides.nie.edu.sg/c.php?g=965796&p=7017302