

**MINI PROJECT**

**MCTA 3202**

GROUP F

GROUP MEMBERS MATRIC NO

1. Muhammad Naufal bin Mohammad Bakri 2110333
2. Muhammad Khairul Aiman bin Mohamad Basari 2113571
3. Muhammad Nabil Ikhmal Bin Khairil Rizal 2114577
4. Muhammad Zakwan Zikri bin Irwan Affandi 2111033
5. Che Muhammad Danial Bin Che Suhaimi 2119075

DATE OF SUBMISSION :

22nd January 2024

**Table of Contents**

## 

[**Abstract 3**](#_895omhk3msch)

[**Introduction 3**](#_ao2fv7vpcfq3)

[**Materials and Equipment 4**](#_g941gudcypw5)

[**Experimental Setup 5**](#_j3ljcad4ojh)

[**Methodology 6**](#_a2gsd1tlvyix)

[**Safety Issues and Precaution 6**](#_trh4wtgm5b0k)

[**Result 7**](#_aq7lie7o1qii)

[**Safety Issue & Safety Precaution 9**](#_wlxuw3c8rqpk)

[**Discussion 9**](#_89b52bjz24m3)

[**Conclusion 10**](#_2ywjb614l6vp)

[**Recommendations 10**](#_iu1zm3qs31z9)

[**References 10**](#_o4vgsbxvmw6s)

[**Appendices 11**](#_cyabo8bzazec)

[**Code Snippets 11**](#_9anlq5n7zg)

[**Student's Declaration 31**](#_9efyym5v108d)

## 

# **Abstract**

In the ambitious pursuit of emulating the functionalities of a genuine washing machine, our project skillfully intertwines the capabilities of an Arduino Mega with a varied array of sensors, encompassing RFID, Water Sensor , and infrared (IR), alongside an assortment of actuators including a DC motor, buzzer, LED, and LCD display.

The implementation of these diverse sensors and actuators give us a quite hard challenge during the design phase of our embedded system device. However, overcoming this challenge has proven to be a pivotal aspect of our project, serving as an invaluable practice session. Not only has it allowed us to navigate the intricacies of system integration, but it has also provided a platform for refining our skills in creating a visually appealing and efficient embedded system.

Our project's thorough approach, especially in integrating RFID for better user interaction and incorporating Water Sensor and IR Sensors to improve efficiency and safety, reflects the challenges found in actual industrial settings. Tackling these diverse aspects in our project serves as excellent training for dealing with the intricacies of industrial environments. In the real world, industries require smooth coordination among different components, much like what we're learning to do in our project. This prepares us well for the complexities and demands of working in industries, where the ability to integrate various elements seamlessly is crucial for achieving success.

# **Introduction**

The objective of this lab is to design and build a basic washer machine prototype using Arduino. This exercise will provide students with insights into the system requirements, hardware components, and programming necessary for creating a simple embedded system device.

For our project, we have designed it so users are able to authenticate with valid RFID cards, enabling an array of options for wash cycles they desire for their garments. Once they have chosen an option out of the modes offered, the user will fill up the tub with clothes. To ensure an effective washing cycle, we have placed an IR sensor to detect if the garments have been piled up to a suitable height, then alerting the user if they have exceeded the limit.

Once the IR sensor has stopped warning the user from overload, water will start filling in the tub to a suitable volume then is limited by a water sensor to alert the system when the water is already filled and ready for the washing process.

In this project, we prioritise user engagement, effective error handling, as well as robust safety protocol, creating an immersive simulation of a fully operational washing machine mechanism.

# **Materials and Equipment**

Identify the equipment, components and materials.

* 2x Arduino Mega 2560 Rev3
* 1x Arduino Pro Micro
* 1x Infrared Sensor Module
* 1x L298N Motor Driver
* 1x Micro DC Motor
* 1x RC522 RFID Module
* 1x Water Level Sensor
* 2x USB Cable For Arduino (A-B Type)
* 1x Piezo Buzzer
* 4-Digit Common Cathode 7-Segment Display
* 16x2 LCD Display
* OLED I2C 0.96Inch 128x64 Blue Display
* 1x Potentiometer
* 3x Push Buttons
* 3x 10kΩ Resistors
* Jumper Wires

# 

# **Experimental Setup**

Circuit Diagram

# 

# 

# **Methodology**

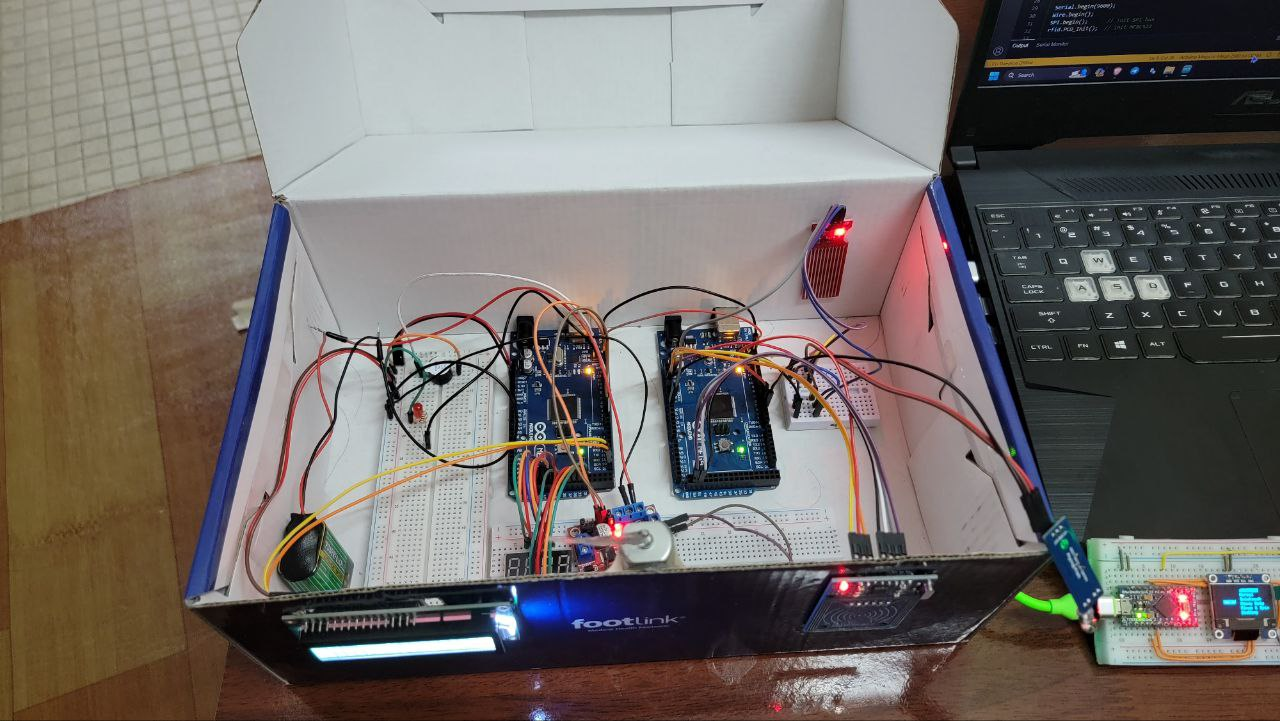
Hardware and Software Setup:

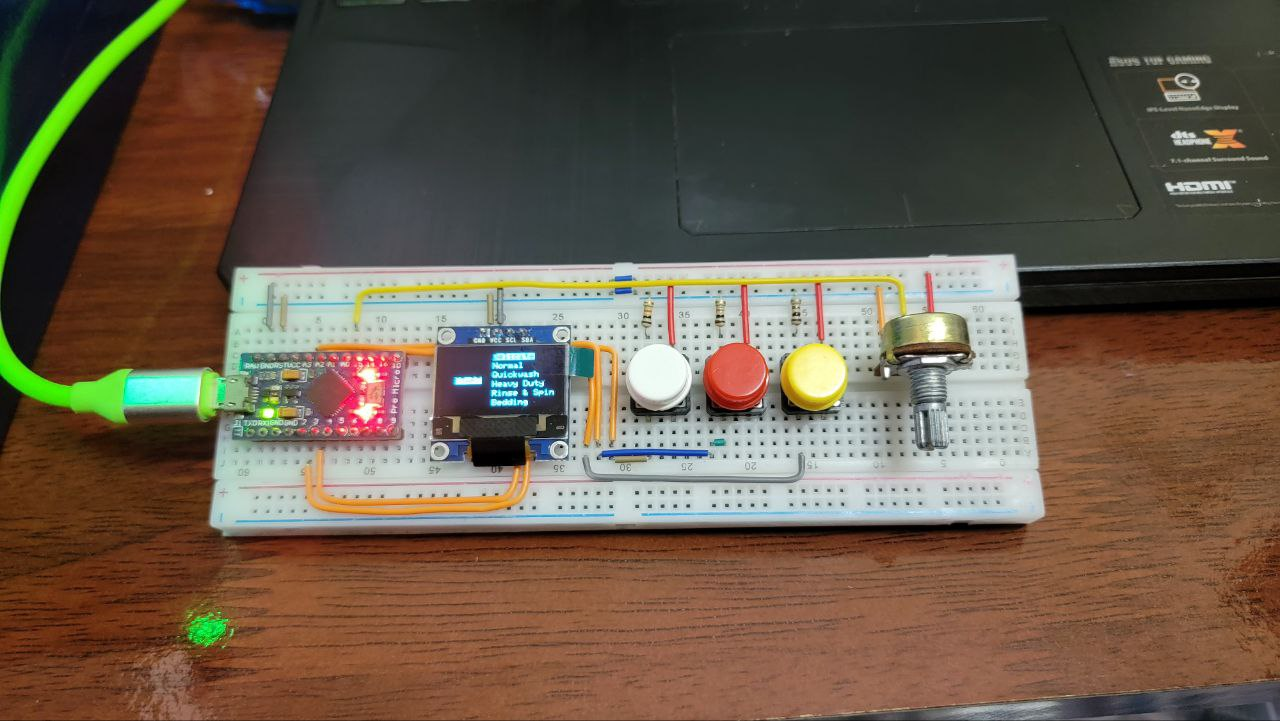
* Connect the two Arduino Mega via Wire Communication(I2C Communication Protocol) for master-slave relationship
* Connect the Master Arduino Mega with the Arduino Pro Micro via Soft Serial Communication Protocol (using python)
* Assemble the rest of the components as shown in the circuit diagram
* Compile and upload the Arduino code in all three microprocessors used

User Process:

* RFID card with authorised UID is needed to be tapped on the RFID reader. Unauthorised cards will not be able to proceed to the next step.
* After authorisation, the user can pick different modes of washing presets using the potentiometer. The presets can be shown on the OLED screen.
* An IR sensor will then detect if the user overloaded the washing machine. The buzzer will go off and the LCD will display “Overload!”. This also prevents the user from starting the washing process.
* After a washing preset has been chosen, there will be a 5 second timer that lets the user make a last-minute decision.
* Then water will then be filling up the drum and triggering the humidity sensor.
* The motor will then start to rotate periodically.
* After the timer of the preset runs out, the motor will stop rotating.

# **Result**





<https://github.com/NotLafuan/GROUP-F-MCTA-3203/raw/main/Mini%20Project/Group%20F%20Project%201.mp4>

<https://github.com/NotLafuan/GROUP-F-MCTA-3203/raw/main/Mini%20Project/Group%20F%20Project%202.mp4>

<https://github.com/NotLafuan/GROUP-F-MCTA-3203/raw/main/Mini%20Project/Group%20F%20Project%203.mp4>

# **Safety Issues and Precaution**

Safety is a crucial aspect when it comes to using household appliances, and washing

machines are no exception.Current safety issues with washing machines include the potential for electrical hazards due to faulty wiring or power cords, water-related concerns such as leaking hoses and incorrect installation, user interactions like child safety and proper loading to prevent imbalance.In this project there are several safety features that we implement to ensure the safety of the user.

The first safety feature that we include is the emergency manual button, a common safety feature found in many machines and mechanisms. Primarily designed for emergency situations, this button serves a crucial role in ensuring safety. Positioned at the control panel for easy accessibility during emergencies, the emergency manual button becomes instrumental in scenarios such as power failures , malfunctions, electrical shock and where there is any water leakage in the machine.When activated, the emergency manual button swiftly disengages the motor that responsible for controlling the rotation of the drum.

Next, we also implement an overload detector in our washing machine prototype.This safety feature is designed to prevent damage to the machine and its components caused by overloading and also to ensure the effectiveness of the cleaning process. Overloading occurs when the laundry load exceeds the maximum weight capacity of that washing machine.We design an overload detector by using an IR sensor to helps protect the washing machine from potential issues such as imbalance, excessive wear and tear, and motor strain.When ever the user load the cloth more than the specified capacity, the buzzer will make a sound and LED will light up in order to alert and warn the user to remove some of the cloth.

Other than that , we also implemented the “Start Delay” mechanism in our washing machine prototype. A start delay helps prevent sudden and unexpected starts of the motor. This is particularly important for motors with high power or those connected to heavy machinery where an abrupt start could be dangerous.This safety features is also significant to reduce the likelihood of electrical surges when starting the motor.This is especially relevant for motors that draw a significant amount of current upon startup like a washing machine.Same just like in other motorised systems, a start delay in a washing machine also helps to reduce the mechanical stress on the motor and other components during startup. This can contribute to the overall longevity of the washing machine.Finally,by implementing Start Delay in our prototype we can overcome the child safety issue because a time delay can provide an added layer of safety by allowing parents to load the machine and set it to start after the children are out of the laundry area.

# **Discussion**

In addressing the identified problem with the humidity sensor, our approach involved mapping the values obtained from the sensor within a range of 1 to 2 to discern between dry and wet conditions. However, during the course of our investigation, we encountered a challenge in reliably detecting when the sensor was wet. Subsequently, we opted to bypass the mapped values and instead focused on utilising the raw sensor data. By analysing the raw values, we were able to identify a specific range that reliably indicated wet conditions, thereby resolving the difficulty associated with detecting sensor wetness. This adjustment allowed for a more accurate and effective determination of the sensor's status, enhancing the overall functionality of the system.

The encountered issue with the LCD display, characterised by glitches and erratic behaviour, led us to investigate a potential connection to a clogged I2C bus. Considering that both the LCD display and the communication between Arduino devices rely on the I2C protocol, it seemed plausible that congestion on the bus was causing the display anomalies. To address this problem, we implemented a solution wherein data transmission on the I2C bus occurs only when there is an update to the information, rather than sending data periodically. This strategic adjustment not only alleviated the strain on the I2C bus but also proved to be an effective resolution to the LCD display glitches, ensuring smoother communication and improved overall system stability.

Moreover, we have initially used a relay to permit a small amount of electrical current for our motor, however a relay is not capable of controlling the speed of the used motor to simulate various processes in a washing machine (wash, rinse, or spin). Hence we have chosen to change to a motor driver to control the velocity as well as the direction of the motor according to our program.

# **Conclusion**

In conclusion, by utilising I2C and Serial Communication between three different Arduinos, we are able to execute various tasks across the integrated system.

From this project, we are able to fully take advantage of the knowledge that is gained throughout the System Integration Lab for this semester. Every single input from the control panel of our Pro Mini and Master Mega is able to communicate with each other effectively. And they are also great at sending the inputs from the user to the Slave Mega to be delivering the programmed output.

This course has helped us a lot in integrating 3 individual systems into 1, and also being able to work interactively and seamlessly with one another.

# **Recommendations**

Currently, the IoT implementation is using the laptop to create a web server to monitor the current state of the washing machine. For the future plan, it is better to use a low powered Single-Board Computer (SBC) like Raspberry Pi board for better energy efficiency and more compact design.

The current implementation of the web server can only be used in the same network. To make it available to be used outside, we plan to use a GSM module where the washing machine will contact the user that the washing cycle is done through SMS or Whatsapp.

# **References**

<https://youtu.be/HVHVkKt-ldc?si=qU-05Reyo7aXp03x>

<https://youtu.be/9HivniieLvI?si=nvSztuzCYWEjwvOq>

# 

# **Appendices**

# Code Snippets

## control/control.ino

#include <SPI.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#include "config.h"

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

String modes[] = {"Delicate", "Normal", "Quickwash", "Heavy Duty", "Rinse & Spin", "Bedding"};

int modeTimers[] = {4 \* 60, 2 \* 60, 20, 5 \* 60, 65, 5 \* 60};

int modeSelector = 0;

enum statuses

{

Stop,

Running,

Pause

};

statuses status; // 0=stop, 1=stop, 2=pause

bool button1Released = true;

bool button2Released = true;

bool button3Released = true;

int timer;

unsigned long startTime;

bool authorized = true;

void potUpdate();

void buttonUpdate();

void displayUpdate();

void displayTime();

void displayModes();

void timerUpdate();

void serialSend();

bool checkUpdate();

void displayNotAuthorized();

void setup()

{

Serial.begin(9600);

status = Stop;

startTime = millis();

pinMode(BUTTON1, INPUT);

pinMode(BUTTON2, INPUT);

pinMode(BUTTON3, INPUT);

pinMode(POT, INPUT);

while (!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS))

;

}

void loop()

{

if (Serial.available())

{

char data = Serial.read();

if (data == 'a')

authorized = true;

else if (data == 'n')

authorized = false;

}

if (authorized)

{

potUpdate();

buttonUpdate();

displayUpdate();

timerUpdate();

}

else

{

displayNotAuthorized();

}

serialSend();

}

void potUpdate()

{

if (status != Stop)

return;

int pot = analogRead(POT);

modeSelector = pot > 170 \* 0 ? 0 : modeSelector;

modeSelector = pot > 170 \* 1 ? 1 : modeSelector;

modeSelector = pot > 170 \* 2 ? 2 : modeSelector;

modeSelector = pot > 170 \* 3 ? 3 : modeSelector;

modeSelector = pot > 170 \* 4 ? 4 : modeSelector;

modeSelector = pot > 170 \* 5 ? 5 : modeSelector;

}

void buttonUpdate()

{

int button1 = digitalRead(BUTTON1);

int button2 = digitalRead(BUTTON2);

int button3 = digitalRead(BUTTON3);

if (button1 && button1Released)

{

button1Released = false;

if (status == Running || status == Pause)

{

status = Stop;

}

else if (status == Stop)

{

status = Running;

}

}

if (button2 && button2Released)

{

button2Released = false;

if (status == Running)

{

status = Pause;

}

else if (status == Pause)

{

status = Running;

}

}

if (button3 && button3Released)

button3Released = false;

if (!button1)

button1Released = true;

if (!button2)

button2Released = true;

if (!button3)

button3Released = true;

}

void displayUpdate()

{

display.clearDisplay();

display.setTextColor(BLACK, WHITE);

display.setCursor(12, 28);

if (status == Stop)

{

display.println("START");

displayModes();

timer = modeTimers[modeSelector];

}

else if (status == Running)

{

display.println("STOP");

displayTime();

}

else if (status == Pause)

{

display.println("PAUSE");

displayTime();

}

display.display();

}

void displayTime()

{

display.setCursor(50, 15);

display.setTextColor(WHITE, BLACK);

display.setTextSize(5);

if (int(timer / 60))

{

char buffer[3];

sprintf(buffer, "%2d\0", int(timer / 60));

display.println(buffer);

display.setTextSize(1);

display.setCursor(110, 42);

display.setTextColor(BLACK, WHITE);

display.println("min");

}

else

{

char buffer[3];

sprintf(buffer, "%2d\0", timer);

display.println(buffer);

display.setTextSize(1);

display.setCursor(110, 42);

display.setTextColor(BLACK, WHITE);

display.println("sec");

}

display.setTextSize(1);

}

void displayModes()

{

int i = 0;

for (String mode : modes)

{

display.setTextSize(1);

if (modeSelector == i)

display.setTextColor(BLACK, WHITE);

else

display.setTextColor(WHITE, BLACK);

display.setCursor(55, 11 \* i);

display.println(mode);

i++;

}

}

void timerUpdate()

{

if (millis() - startTime > 1000)

{

if (button3Released)

startTime = millis();

if (status == Running)

timer--;

if (timer <= 0)

status = Stop;

}

}

void serialSend()

{

Serial.print(status);

Serial.print(" ");

Serial.print(timer);

Serial.print(" ");

Serial.println(modes[modeSelector]);

}

void displayNotAuthorized()

{

display.clearDisplay();

display.setTextColor(WHITE, BLACK);

display.setCursor(15, 10);

display.setTextSize(2);

display.println("Card Not");

display.print("Authorized");

display.setTextSize(1);

display.display();

}

## control/config.h

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define OLED\_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)

#define SCREEN\_ADDRESS 0x3C

#define BUTTON1 A1

#define BUTTON2 A2

#define BUTTON3 A3

#define POT A0

#define THIRTY\_MINUTES 30\*60

## master/master.ino

#include <SPI.h>

#include <MFRC522.h>

#include <Wire.h>

#include "config.h"

MFRC522 rfid(SS\_PIN, RST\_PIN);

MFRC522::MIFARE\_Key key;

byte nuidPICC[4]; // Init array that will store new NUID

byte uid1[4] = {0x23, 0x0C, 0xC3, 0x10}; // authorize card1

byte uid2[4] = {0x23, 0x1F, 0xE6, 0x0D}; // authorize card2

bool verify;

bool motor = false;

char outgoingByteIR;

char outgoingByteWaterLevel;

char outgoingByteCard;

char outgoingByteMotor;

int sendData = 0;

void setup()

{

pinMode(IR, INPUT);

pinMode(WATER\_LEVEL\_PIN, INPUT);

Serial.begin(9600);

Wire.begin();

SPI.begin();

rfid.PCD\_Init();

for (byte i = 0; i < 6; i++)

key.keyByte[i] = 0xFF;

}

void loop()

{

if (rfid.PICC\_IsNewCardPresent())

{

rfid.PICC\_ReadCardSerial();

for (byte i = 0; i < 4; i++)

nuidPICC[i] = rfid.uid.uidByte[i];

// printHex(rfid.uid.uidByte, rfid.uid.size);

// Serial.println();

for (size\_t i = 0; i < 4; i++)

{

if (nuidPICC[i] == uid1[i] || nuidPICC[i] == uid2[i])

verify = true;

else

{

verify = false;

break;

}

}

}

if (Serial.available())

{

char data = Serial.read();

if (data == 'g')

motor = true;

else if (data == 'h')

motor = false;

}

if (verify)

outgoingByteCard = 'n';

else

outgoingByteCard = 'a';

if (digitalRead(IR))

outgoingByteIR = 'o';

else

outgoingByteIR = 'k';

if (analogRead(WATER\_LEVEL\_PIN) >= 250)

outgoingByteWaterLevel = 'f';

else

outgoingByteWaterLevel = 'e';

if (motor)

outgoingByteMotor = 'g';

else

outgoingByteMotor = 'h';

Serial.print(outgoingByteIR);

Serial.print(" ");

Serial.print(outgoingByteWaterLevel);

Serial.print(" ");

Serial.print(outgoingByteCard);

Serial.print(" ");

Serial.println(analogRead(WATER\_LEVEL\_PIN));

if (millis() - sendData >= 500)

{

sendData = millis();

Wire.beginTransmission(9);

Wire.write(outgoingByteIR);

Wire.write(outgoingByteWaterLevel);

Wire.write(outgoingByteCard);

Wire.write(outgoingByteMotor);

Wire.endTransmission();

}

}

void printHex(byte \*buffer, byte bufferSize)

{

for (byte i = 0; i < bufferSize; i++)

{

Serial.print(buffer[i] < 0x10 ? " 0" : " ");

Serial.print(buffer[i], HEX);

}

}

## master/config.h

#define IR 3

#define WATER\_LEVEL\_PIN A0

// for RFID

#define SS\_PIN 53

#define RST\_PIN 5

## slave/slave.ino

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <SevSeg.h>

#include "config.h"

SevSeg sevseg;

unsigned long motorTimer;

bool wash = false;

int timer = 6000;

int prevTimer = 0;

bool update = false;

bool motorDir = false;

unsigned long motorDirStart = 0;

char water, load, card, motor;

char prevWater, prevLoad, prevCard;

int in1 = 2;

int in2 = 3;

char test;

LiquidCrystal\_I2C lcd(0x27, 16, 2);

void receiveEvent(int);

void setup()

{

Serial.begin(9600);

Wire.begin(9); // Start the I2C Bus as Slave on address green

Wire.onReceive(receiveEvent); // Attach a function to trigger when something is received.

lcd.init();

lcd.backlight();

pinMode(RED, OUTPUT);

pinMode(BUZZER, OUTPUT);

pinMode(in1, OUTPUT);

pinMode(in2, OUTPUT);

digitalWrite(in1, LOW);

digitalWrite(in2, LOW);

sevseg.begin(HARDWARECONFIG, numDigits, digitPins, segmentPins, RESISTORSONSEGMENTS,

UPDATEWITHDELAYS, LEADINGZEROS, DISABLEDECPOINT);

sevseg.setBrightness(90);

}

void loop()

{

if (wash)

{

timer = motorTimer - millis();

timer = timer / 1000;

if (timer != prevTimer && timer >= 0)

{

prevTimer = timer;

sevseg.setNumber(timer, 1);

}

}

if (motor == 'g')

{

if (!wash && water == 'f')

{

wash = true;

motorTimer = millis() + 6000;

}

if (water == 'e')

wash = false;

}

else

{

wash = false;

timer = 6000;

}

if (card == 'a')

{

if (update)

{

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Card");

lcd.setCursor(0, 1);

lcd.print("Authorized");

}

if (load == 'k')

{

if (update)

{

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Overload!");

}

wash = false;

digitalWrite(RED, HIGH);

digitalWrite(BUZZER, HIGH);

}

if (load == 'o')

{

digitalWrite(RED, LOW);

digitalWrite(BUZZER, LOW);

}

}

if (card == 'n' && update)

{

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Card Not");

lcd.setCursor(0, 1);

lcd.print("Authorized");

}

if (timer <= 0 && wash)

spinMotor();

else

stopMotor();

detectUpdate();

sevseg.refreshDisplay();

}

void spinMotor()

{

if (millis() - motorDirStart >= 3000)

{

if (motorDir)

motorDir = false;

else

motorDir = true;

motorDirStart = millis();

}

if (motorDir)

{

analogWrite(in1, 100);

digitalWrite(in2, LOW);

}

else

{

digitalWrite(in1, LOW);

analogWrite(in2, 100);

}

}

void stopMotor()

{

digitalWrite(in1, LOW);

digitalWrite(in2, LOW);

}

void receiveEvent(int bytes)

{

load = Wire.read();

water = Wire.read();

card = Wire.read();

motor = Wire.read();

}

void detectUpdate()

{

if (card != prevCard)

{

update = true;

prevCard = card;

}

else if (load != prevLoad)

{

update = true;

prevLoad = load;

}

else if (water != prevWater)

{

update = true;

prevWater = water;

}

else

{

update = false;

}

}

## slave/config.h

#include <SevSeg.h>

#define RED 6

#define BUZZER 7

byte numDigits = 4;

byte digitPins[] = {52, 46, 44, 31};

byte segmentPins[] = {50, 42, 35, 39, 41, 48, 33, 37};

#define RESISTORSONSEGMENTS false

#define HARDWARECONFIG COMMON\_ANODE

#define UPDATEWITHDELAYS false

#define LEADINGZEROS true

#define DISABLEDECPOINT true

## washing\_machine.py

import serial

import requests

import threading

import time

data\_control = ''

data\_master = ''

thread\_stop = False

card = ''

status = '1'

timer = 1

mode = 1

def update\_control():

global thread\_stop, data\_control, data\_master

with serial.Serial('COM16', 9600) as ser:

while not thread\_stop:

data\_control = ser.readline().decode().strip()

ser.write(card.encode())

def update\_master():

global thread\_stop, data\_master, status

with serial.Serial('COM24', 9600) as ser:

while not thread\_stop:

data\_master = ser.readline().decode().strip()

if status == '1':

ser.write('g'.encode())

else:

ser.write('h'.encode())

def update\_web():

global thread\_stop, data\_control, status, timer, mode

while not thread\_stop:

post = {"status": status, "timer": timer, "mode": mode}

requests.post('http://127.0.0.1:5000/send\_data', json=post)

time.sleep(1)

if \_\_name\_\_ == '\_\_main\_\_':

control\_thread = threading.Thread(target=update\_control)

master\_thread = threading.Thread(target=update\_master)

web\_thread = threading.Thread(target=update\_web)

control\_thread.start()

master\_thread.start()

web\_thread.start()

try:

while True:

try:

data = f'{data\_control} {data\_master}'

ir, water, card, level = data\_master.split()

status, timer, \*mode = data\_control.split()

mode = ' '.join(mode)

print(f'\r{data:100}', end='')

except ValueError:

...

except KeyboardInterrupt:

...

except Exception as e:

print(e)

thread\_stop = True

print('Done')

## webserver.py

from flask import Flask, request, jsonify, render\_template

app = Flask(\_\_name\_\_)

data = {"status": 0,

"timer": 0,

"mode": ""}

@app.route('/')

def hello\_world():

global data

return render\_template('base.html')

@app.route('/send\_data', methods=['POST'])

def send\_data():

global data

data = request.json

return jsonify(data)

@app.route('/get\_status', methods=['GET'])

def get\_status():

global data

return data['status']

@app.route('/get\_timer', methods=['GET'])

def get\_timer():

global data

return data['timer']

@app.route('/get\_mode', methods=['GET'])

def get\_mode():

global data

return data['mode']

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', debug=True)

## templates/base.html

<!DOCTYPE html>

<html lang="en">

<link rel="shortcut icon" href="{{ url\_for('static', filename='favicon.ico') }}">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Washing Machine</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.css">

<script src="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.js"></script>

<script src="static/js/update\_status.js" defer></script>

</head>

<body>

<div style="margin-top: 50px;text-align: center">

<span id="status" class="ui gray label">start</span>

<span id="timer" class="ui gray label">20</span>

<span id="mode" class="ui gray label">Normal</span>

</div>

</body>

</html>

## static/js/update\_status.js

function pad(num, size) {

num = num.toString();

while (num.length < size) num = "0" + num;

return num;

}

async function updateStatus() {

const spans = document.querySelectorAll('span');

while (true) {

try {

spans.forEach(async span => {

console.error(span.id);

if (span.id == "status") {

const response = await fetch('/get\_status');

const data = await response.text();

if (data == 0) {

span.textContent = "Stopped";

span.classList.remove('gray');

span.classList.remove('green');

span.classList.add('red');

} else if (data == 1) {

span.textContent = "Running";

span.classList.remove('gray');

span.classList.add('green');

span.classList.remove('red');

} else if (data == 2) {

span.textContent = "Paused";

span.classList.add('gray');

span.classList.remove('green');

span.classList.remove('red');

}

} else if (span.id == "timer") {

const response = await fetch('/get\_timer');

const data = await response.text();

const timer = parseInt(data);

const minutes = Math.floor(timer / 60)

const seconds = timer % 60

span.textContent = minutes.toString() + ":" + pad(seconds, 2);

} else if (span.id == "mode") {

const response = await fetch('/get\_mode');

const data = await response.text();

span.textContent = data;

}

})

await new Promise(resolve => setTimeout(resolve, 500));

} catch (error) {

console.error('An error occurred:', error);

}

}

}

updateStatus();

# **Student's Declaration**

**Certificate of Originality and Authenticity**

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual’s contributors to the report.

We therefore, agreed unanimously that this report shall be submitted for marking and this final printed report has been verified by us.

Signature: Naufal Read

Name: Naufal Understand

Matric Number: 2110333 Agree

Contribution: Testing and Debugging

Signature: Aiman Read

Name: Aiman Understand

Matric Number: 2113571 Agree

Contribution : Programming

Signature: Nabil Read

Name: Nabil Understand

Matric Number: 2114577 Agree

Contribution: Circuit Design

Signature: Zakwan Read

Name: Zakwan Understand

Matric Number: 2111033 Agree

Contribution: Data Analysis

Signature: Che Read

Name: Che Understand

Matric Number: 21119075 Agree

Contribution : Discussion