EE344 : Design Report

Enhanced Features for Corrosion Monitoring Product designed at WEL (iPEC) HMI System

TUE-25 group

Prajwal Kalpande , 200070028
Jaideep Kotani , 200070035
S.V.Sai Siddartha, 200070074

Project Guides :
Prof. Siddarth Tallur
Prof. Anil Kottantharayil

RAs/TAs :
Srinidhi
Madhusudan

Department of Electrical Engineering
Indian Institute of Technology Bombay

Contents

1 Introduction

1.1 Aim

The project aims to develop a Human Machine Interface (HMI) system which can be used with a corrosion monitoring product(iPEC-Probe). The HMI system will include a user interface with buttons that will be connected to a microcontroller. Since we can't connect this HMI system with iPEC Probe as of now, various sensors have been used to gather data that will be displayed on the LCD Screen.

1.2 Description of the Project

In this project, we will use a Raspberry Pi Pico microcontroller board as the central controller for the HMI system. The microcontroller board will be responsible for interfacing with the various buttons and the LCD screen. The buttons will be connected to the microcontroller board and used to interact with the LCD screen. The microcontroller board will then control the display on the LCD screen based on the input from the buttons.

The HMI system displays the quantities measured by the iPEC probe in real time, including the distance travelled from the scanning start point, rebar depth, rebar size, and other parameters. It allows the user to have a real-time overview of the measurements being taken and make informed decisions about the condition of the concrete structure. As mentioned earlier, we have obtained values from various sensors instead of the quantities mentioned above; otherwise, we would have had to use dummy fixed values on the screen. This helps in comprehensive testing of the HMI as values from the sensor will vary, like if it's connected with the iPEC Probe.

2 Design

2.1 Block Diagram

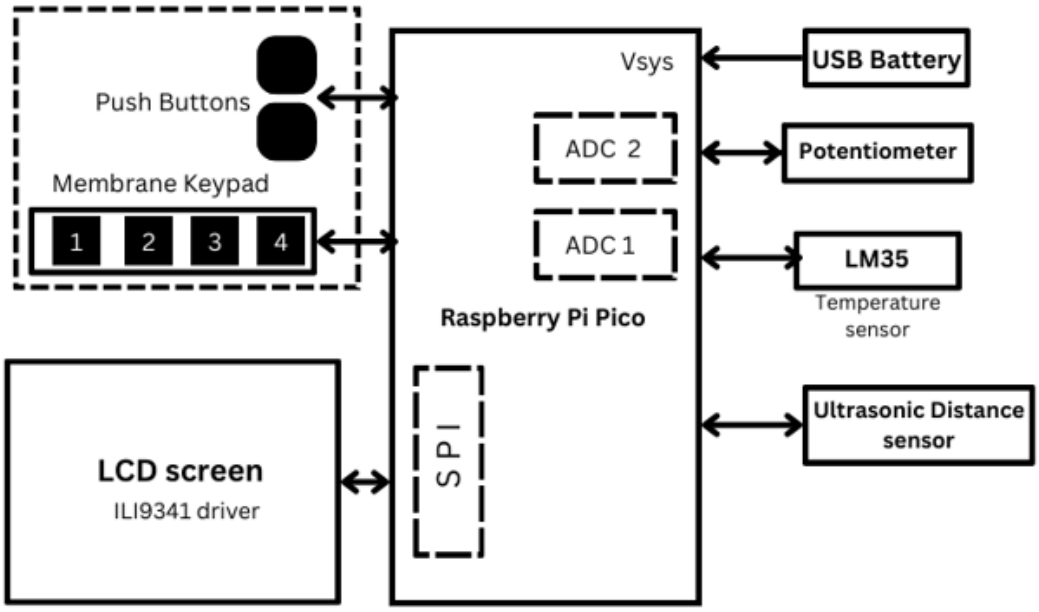


Figure 1: HMI System Block Diagram

2.2 Principle of Operation of each Subsystem

2.2.1 LCD Display

LCD uses individual pixels to display data or images. Each pixel displays a RGB color value. The microcontroller powers the LCD through a 3.3V supply. The switches are connected to the microcontroller board and are used to interact with the LCD. The microcontroller board controls the display on the LCD screen based on the input from the buttons by sending appropriate data and commands via SPI. The user interface which we displayed on the LCD screen is as given below:

2.2.2 User Interface

We have six switches, out of which two are push buttons and four are from membrane keypad. Let p1 and p2 be the names of push buttons and m1, m2, m3, and m4 are the names of membrane keypad switches as mentioned on them

- m1 switch represents the left arrow on the screen, using which we can go to the previous screen
- m2 switch represents the right arrow on the screen, using which we can go to the next screen
- m3 is the power switch, which will be used to turn the device on/off
- m4 is Esc/Home, using which we can close the pop-up window/dialog box (INFO box) or directly go to the first screen, i.e., distance screen, by pressing it.
- p1 switch is the info button; when we press the p1, there will be a pop-up window containing user guidelines on how to use the device.

- p2 switch is for stopping/starting the simulation screen, which is displaying how deep the rebars are in the wall

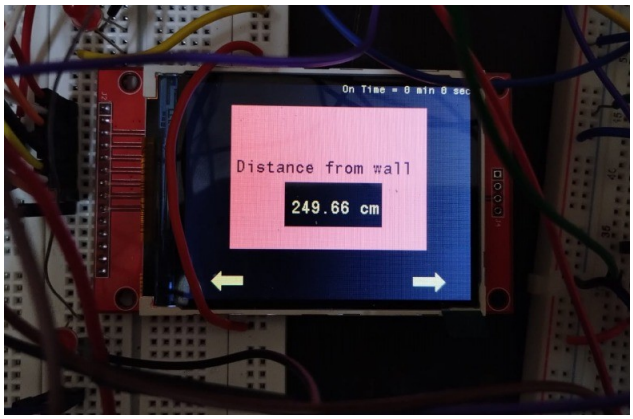


Figure 2: Distance from Ultrasonic Sensor

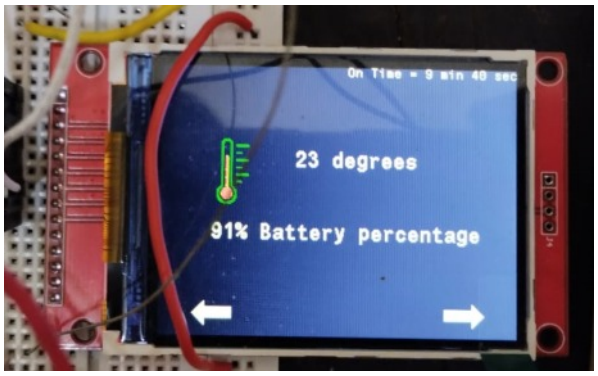


Figure 3: Room Temperature from LM35

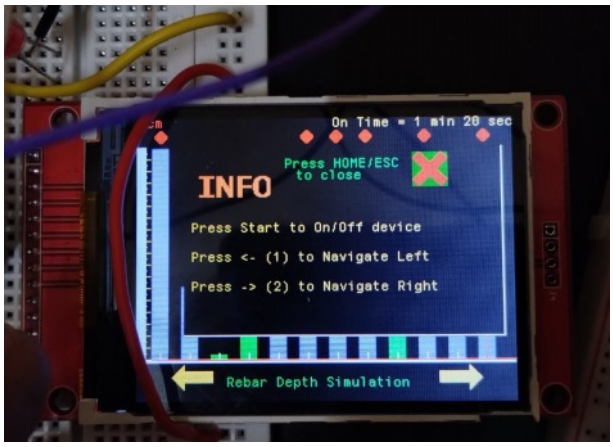


Figure 4: Info Screen 1

Figure 5: Info Screen 2

Figure 6: Screen Brightness

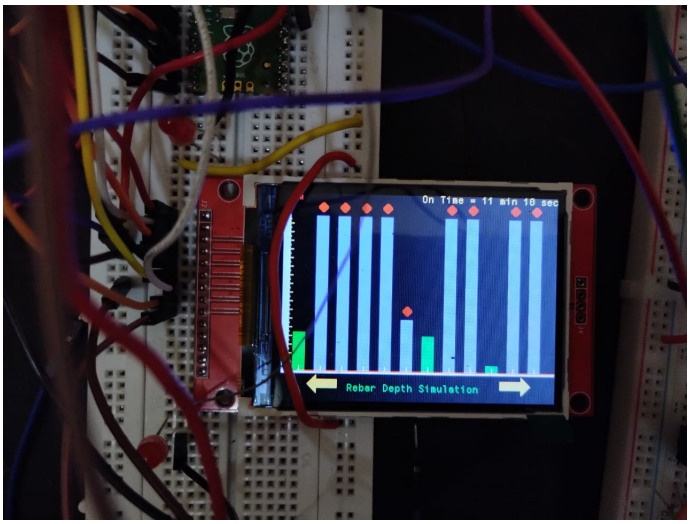


Figure 7: Depth Simulation

Fig.7 is just a simulation of the iPEC design. It will show how much depth the rebar is inside the wall; after some value, it will display nothing if no rebar is there. We didn't have the iPEC,

so we will show this dummy frame where the bars represent distances measured by the ultrasonic distance sensor at regular intervals. The rightmost visible bar is the latest measured one. The height of the bars is proportional to their distance from the wall. Red circles are indicated at time instants where the distance exceeds a certain threshold.

If we press m3 (OK button) and currently LCD displays Rebar Simulation, the simulation gets paused, and the bars don't move left since no measurements are being made until one presses the OK button again. After which distance is measured, that bar becomes the rightmost, and all those before being shifted to the left. This can be understood better from the video demo attached at the end.

On the screen displaying, temperature and battery percentage are also displayed. It isn't the actual battery percentage. Based on the time the device has been on, we update it appropriately(as of now, it is just 100 - no. of minutes passed).

The screen for adjusting the brightness will display horizontal bars and a number indicating the exact brightness of the LCD. For testing, we also printed the difference between the previous brightness and that obtained after rotating the potentiometer to visualize better and implement smooth bar animation.

When the dialog box(INFO) opens, one can only navigate left and right along the instructions/info given in that part and not across screens. One must first close the box using Home/Esc(m4), and then one can navigate across different screens as before.

2.2.3 Sensors

Sensors were used to gather data for displaying on the LCD screen. The sensors and HMI are interfaced through an analog-to-digital converter (ADC) that converts the analog signals from the sensors into digital values that the software can process. The microcontroller processes these values and then forwards them to the LCD to display accordingly. The sensors used in this project are:

LM35 temperature sensor

This sensor measures the temperature using the proportional voltage output principle using a solid-state technique. It has a linear scale factor of +10 mV/°C, which means that the output voltage at the Vout pin increases by ten millivolts for every 1-degree Celsius increase in temperature.

The LM35 sensor converts the ambient temperature into electrical voltage by a circuit in the IC, where the temperature change is proportional to the output voltage changes. The sensor can measure temperatures from -55°C to 150°C, and then we interface with the microcontroller using the ADC function.

Ultrasonic Distance Sensor

This sensor measures the distance of the objects from the sensor by emitting ultrasonic waves and converting the reflected waves into electrical signals. It measures the distance by using the following formula:

$$\text{distance} = \text{speed} * \text{time}/2$$

There are four pins for this sensor, they are V_{CC}, GND, TRIG, and ECHO.V_{CC} will be connected to 5V and GND is connected to the ground. TRIG pin will give input to generate the pulse to transmit the waves from the transmitter, now we will start the timer. The ECHO pin will be turned on when the reflected wave is received and we will stop the timer. Using the time from the timer we will measure the distance.

As Raspberry Pi Pico GPIO pins i/o voltage is 3.3V, we will use two 10K resistors in series from ECHO to GND, we use these as a divider to convert the 5V logic level to a safe 2.5V. For Raspberry Pi Pico any voltage between 1.8V and 3.3V will be read by the Raspberry Pi as high and anything lower than 1.8V will be read as low, so no problem with the microcontroller for converting from 5V to 2.5V.

Potentiometer

A 10k Potentimeter connected to 3.3V and GND was used to adjust the voltage supplied to LED pin of LCD screen allowing us to control the brightness of the screen. This value is read by using an onboard ADC and displayed on the LCD.

2.2.4 Switches

Switches control the current flow in a circuit by opening or closing the contacts. The switches are interfaced with the HMI software through a digital input/output (I/O) module that detects the state of the switches and sends the corresponding signals to the software. The switches can be used to control the functions of the HMI, such as turning on/off the display, selecting the sensor to display, or adjusting the settings.
We wrote code for Raspberry Pi Pico using C SDK that checks for input from the switches using interrupts and performs certain actions as explained in User Interface section above. The sensors mentioned above and Raspberry Pi Pico are connected, and their input readings are used to update the UI on LCD.

2.3 Components Selection

2.3.1 TFT LCD Display

Name	TFT SPI Screen Module
Screen Size	2.8 Inch
Driver IC	ILI9341
Touchscreen	No
Pixel Resolution	240x320
Display Type	individual RGB pixel control
supply voltage	3.3 - 5 V
PCB Size	85 x 48 mm

The LCD screen will be used for displaying the user interface, which will be interacted with by the user using buttons. We have tried to mimic the user interface, which is used by corrosion monitoring products by using suitable sensors. Since the device will be used in construction areas where there may be intense sunlight, we need to make sure that the user interface displayed on the screen is visible in bright conditions as well. Moreover, as the screen will be held away from eyesight (when scanning remote regions) wide view angle is preferred.
Keeping this in mind, we cannot obviously use the usual green LCD screens since they have poor contrast,wide view angle and not visible in sunlight. This left us two choices: OLED LCD display and TFT(thin film transistor) LCD Display. Although OLED LCD provide higher contrast than TFT LCD, TFT has better visibility in bright conditions and even direct sunlight which is more important.So we selected a TFT LCD screen for this project.
The display has two modes: SPI and 8-bit. We use the SPI mode with nine pins, whereas the 8-bit mode uses 12 pins for interfacing. Another advantage of the SPI is that we can use the onboard MicroSD card socket to display images if needed.

2.3.2 Potentiometer

We used a 10k potentiometer to adjust the brightness of the LCD screen. The voltage value is read by the microcontroller and the brightness value is displayed on the LCD.

2.3.3 Ultrasonic Distance Sensor

Name	Ultrasonic Sonar Distance Sensor
supply voltage	5 V
return Echo	5 V logic
Current	15mA
Ultrasonic Frequency	40 KHz
Measuring Angle	15°
Trigger Input Signal	10uS high pulse

This sensor uses ultrasonic waves to measure the distance to an object. It sends a high-frequency sound wave to calculate distance and the sensor has a range of 4m with an accuracy of 3mm. Any other distance sensor would have also sufficed but since we are familiar with Ultrasonic sensors and it's easily available we went with it. We have used the distance value obtained to simulate iPEC probe functionality as explained in User Interface section.

2.3.4 LM35 Temperature Sensor

Name	Voltage Analog Temperature Sensor
supply voltage	4 - 30 V
supply current(max)	114 uA
Interface	Analog output
Accuracy	0.5°C at room temperature

LM35 is a precision integrated circuit having an analog output voltage proportional to the temperature. The sensor is interfaced with the microcontroller, and the output values will be processed by onboard ADC and displayed on LCD.

2.3.5 Push Buttons, Membrane Keypad and LEDs

We used push buttons and a membrane keypad to send the interrupt signals to the microcontroller; after receiving an interrupt, the microcontroller performs necessary actions and updates the content on the LCD screen.
For feedback purposes, we have used green LEDs; they turn on when push buttons are pressed and turn off when the pressure is released from the button so that these will confirm whether we successfully gave the interrupt signal to the microcontroller or not.

2.3.6 Passive components

Resistors
We are using two resistors of 10KΩ and one capacitor of 1μF. The divider circuit uses two resistors to make the ultrasonic sensor's output(Echo signal) from 5V to 2.5V. The signal will be high between the time the ultrasonic sound wave is sent and received. As soon as the wave is received back, the Echo signal goes low, i.e. to 0V. As mentioned in the datasheet of raspberry pi pico, the maximum input voltage for GPIO pins is 3.3V, and any higher voltage damages the microcontroller. Also, the GPIO pin is considered high(logic 1) if the input voltage is $\geq$1.8V.

Thus, we need to maintain an input voltage between 1.8 and 3.3V, achieved by our choice of resistor values. Moreover, GPIO pins sink/source current should be$\leq$0.5 mA to avoid damage. In our case, we will get a current of 0.05mA which is very well within the safety limit.

Capacitor

The capacitor is used for maintaining the 3.3V across the circuit in case of any minor fluctuations.

2.4 Similarities and Variations in Project with other group

We have used the same microcontroller board and LCD screen (ours is slightly bigger). The sensors are different: our group used Distance, and Temperature sensor while the other group has used Power measuring module for battery, SD card and Bluetooth module. Further, the user interface, features and use of buttons (for navigation, selecting option, etc.) are also different.

3 Circuit and PCB Design

3.1 Circuit Schematic

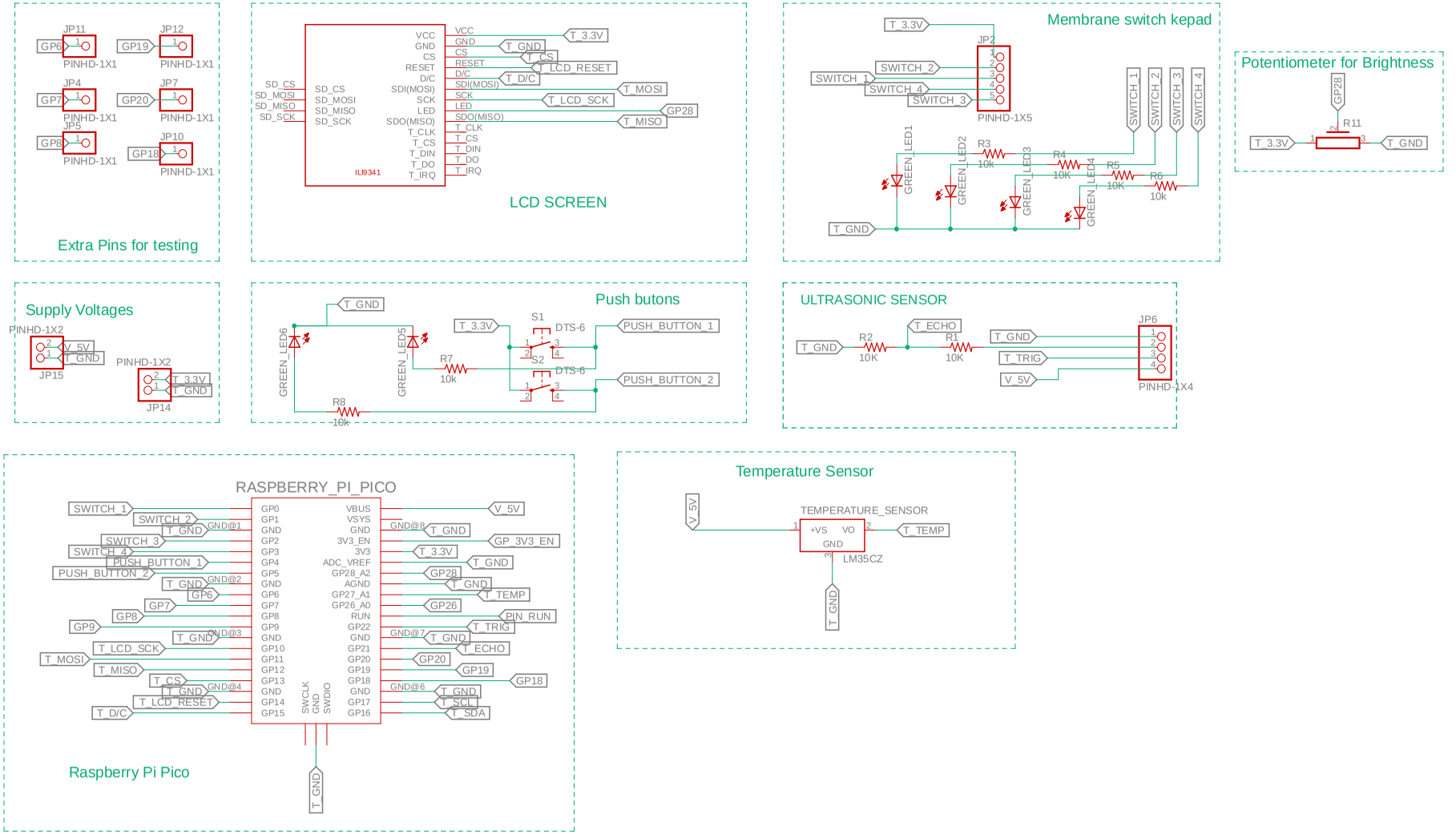
We used Raspberry Pi Pico as a microcontroller. We have an Ultrasonic sensor and a temperature sensor for measuring distance and temperature respectively. The values of these will be displayed on the LCD screen and we can control it by using a pair of push buttons, a 4X1 membrane keypad, and a potentiometer to control the brightness of the LCD screen.

LCD screen uses SPI communication to communicate with the Raspberry PI Pico. It has a total of 9 pins, out of which three are V_{CC}, GND and backlight LED, two are RESET and DC, and the remaining four pins for SPI communication, we used GP10, GP11, GP12, GP13 for SCK, TX, RX, CS respectively.

The Ultrasonic sensor sends the pulses, so we use standard GPIO pins only. We need ADCs to convert the analog voltage value into the digital value for the temperature sensor and potentiometer, so we use two ADC pins for it. The membrane keypad and Push buttons are connected to various GPIO pins, as any GPIO pin can be used as an interrupt in Raspberry Pi Pico, and the LEDs are for feedback purposes.

We also connected the remaining pins of Raspberry Pi Pico to pin headers so that we can use them for testing the microcontroller.

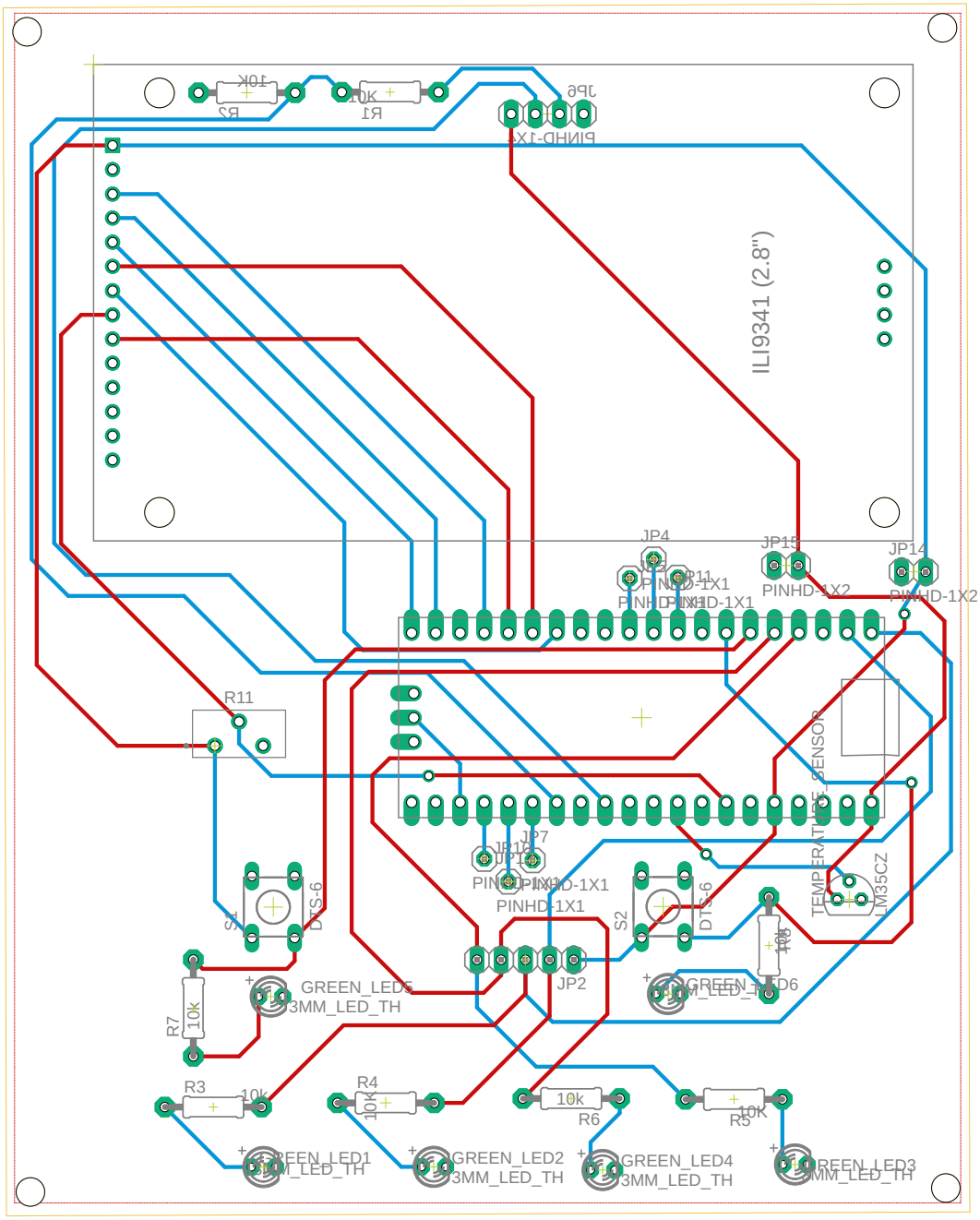
Schematic


Extra Pins for testing
JP11
JP12
JP4
JP7
JP5
JP10
PINHD-1X1
GP6
GP19
GP7
GP20
GP8
GP18
LCD SCREEN
ILI9341
SD_CS
SD_MOSI
SD_MISO
SD_SCK
VCC
GND
CS
RESET
D/C
SDI(MOSI)
SCK
LED
SDO(MISO)
T_CLK
T_CS
T_DIN
T_DO
T_IRQ
T_3.3V
T_GND
T_LCD_RESET
T_D/C
T_MOSI
T_LCD_SCK
GP28
T_MISO
Membrane switch kepad
JP2
PINHD-1X5
SWITCH_1
SWITCH_2
SWITCH_3
SWITCH_4
GREEN_LED1
GREEN_LED2
GREEN_LED3
GREEN_LED4
R3
R4
R5
R6
10K
Potentiometer for Brightness
GP28
R11
Supply Voltages
PINHD-1X2
JP15
JP14
V_5V
Push butons
S1
S2
DTS-6
PUSH_BUTTON_1
PUSH_BUTTON_2
R7
R8
10k
ULTRASONIC SENSOR
JP6
PINHD-1X4
T_ECHO
T_TRIG
R1
R2
Temperature Sensor
TEMPERATURE_SENSOR
LM35CZ
+VS
VO
GND
T_TEMP
RASPBERRY_PI_PICO
Raspberry Pi Pico
GP0
GP1
GP2
GP3
GP4
GP5
GP6
GP7
GP8
GP9
GP10
GP11
GP12
GP13
GP14
GP15
GP16
GP17
GP18
GP19
GP20
GP21
GP22
GP26_A0
GP27_A1
GP28_A2
RUN
ADC_VREF
3V3
3V3_EN
VSYS
VBUS
SWCLK
SWDIO
GP_3V3_EN
PIN_RUN
T_SCL
T_SDA
T_TRIG
T_ECHO
GND@1
GND@2
GND@3
GND@4
GND@6
GND@7
GND@8

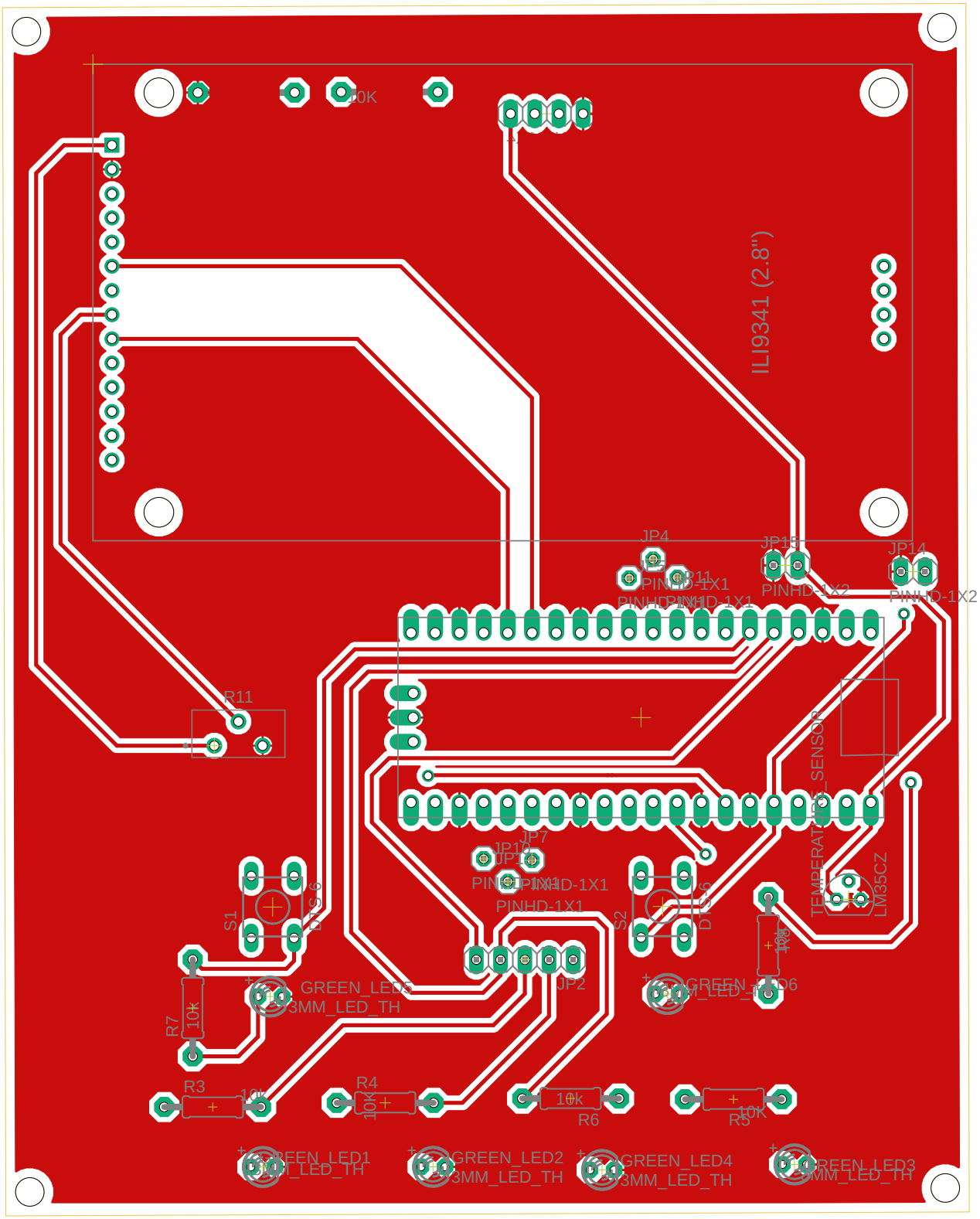
3.2 PCB Layout

We have made the PCB layout such that the LCD screen is at the top and buttons to control it at the bottom. All the components are on the top layer except the ultrasonic sensor and its resistors. The ultrasonic is just below the LCD screen. The raspberry pi pico and the potentiometer are in the middle of PCB layout .

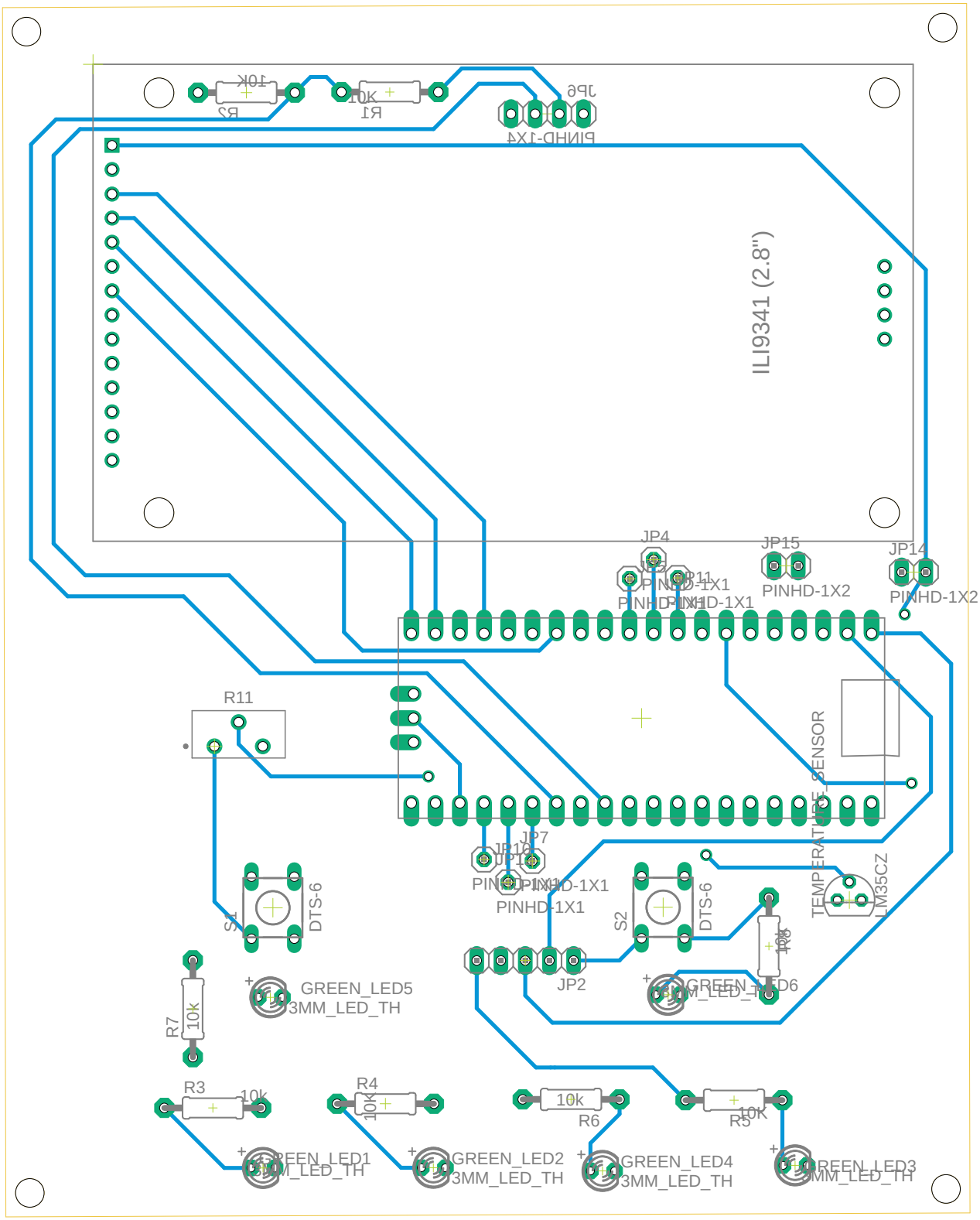
PCB Both Layers



PCB Top Layer



PCB Bottom Layer



4 Breadboard level Testing of all Subsystems

General Setup:
For testing all the sensors and the proper working of the buttons, we used the LCD screen to print information for debugging purposes. Thus, once the code is loaded in the microcontroller, we print appropriate messages on LCD for testing whenever any sensor value changes or the user pushes any button.

All the sensors, buttons, LEDs, passive components (Resistors and capacitors) and LCD screen are interfaced with the microcontroller as given in the schematic on the breadboard using wires and jumper wires. The microcontroller was powered up using USB for testing the subsystems. Detailed testing methodology and results are as follows:

4.1 Microcontroller and software setup

4.1.1 Testing Setup & Method

This involved preliminary microcontroller testing, checking our software setup (for missing libraries and software), compiling basic programs, and dumping them on the microcontroller.

Testing Method
For the compilation of code, we need to use Cmake, which builds the files and generated executables (.uf2) dumped on the Raspberry Pi Pico board. We wrote code to blink the LED on the Raspberry pi Pico board to test whether the microcontroller works correctly and whether the executables work as intended. Since LCD Screen was unavailable then, we tried serial input output using USB.
The Blink LED code is available in Raspberry Pi Pico C SDK documentation which, after compiling and dumping, blinks the onboard LED continuously. Hello World code from documentation was used for the Serial Input Output part, and the data printed was viewed using RealTerm software.

4.1.2 Results:

The LED blinked as expected, and serial I/O also worked properly. There were no errors while building the code.
Note. We didn't need Serial I/O for debugging or other purposes; instead, we printed directly on LCD.

4.2 Temperature Sensor

Overview
The LM35 temperature sensor is a precision integrated circuit that provides an analog output voltage proportional to the temperature. In our system, the sensor is interfaced with the microcontroller, and the output values are processed and displayed on an LCD screen. Testing the subsystem is crucial to ensure that the temperature readings displayed on the screen are accurate and reliable.

4.2.1 Testing Setup & Method

- **Hardware Setup**: We connected the LM35 temperature sensor to the microcontroller using VCC, GND, and output wires. We used a breadboard to make the connections and verified them using a multimeter.

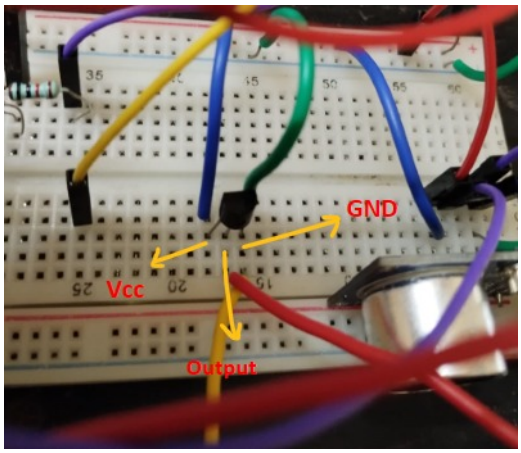


Figure 8: Hardware setup

- **Software Setup**: We wrote a simple program in C language to read the analog voltage output of the LM35 temperature sensor and convert it into temperature values in Celsius. We used the microcontroller's ADC (Analog to Digital Converter) module to read the voltage values.

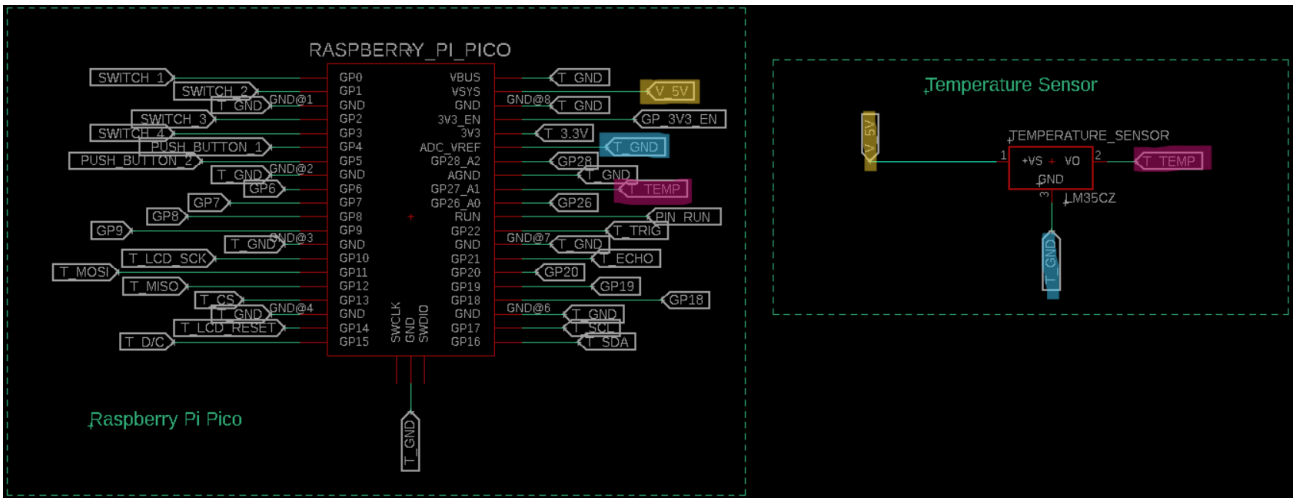


Figure 9: Sensor setup

Testing Method

- To convert the ADC values to temperature values in Celsius, we used a simple mathematical formula based on the LM35's specifications. According to the datasheet, the sensor provides an output voltage of 10mV per degree Celsius.
- We first multiplied the ADC value by the reference voltage of the microcontroller (3.3V) and divided the result by 1024 (number of bits in the ADC). This gave us the voltage value in volts.
- We then multiplied this voltage value by 100 (to convert from volts to millivolts), and divided the result by 10 (to account for the 10mV per degree Celsius output of the LM35). This gave us the temperature value in Celsius.

4.2.2 Testing Results

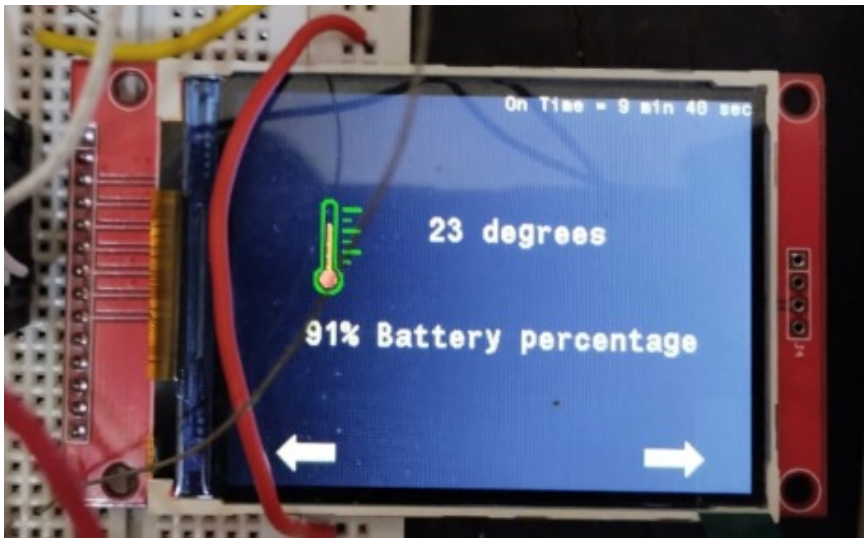


Figure 10: Result

- **Calibration**: We placed the LM35 temperature sensor in a room with a known temperature and verified that the temperature readings displayed on the LCD screen were accurate. We repeated this process for different temperatures to ensure the readings were consistent. We also compared the results obtained from different LM35 sensors to ensure that the conversion method was consistent across different sensors.
- **Integration Testing**: Finally, we integrated the temperature sensor subsystem with the rest of the system and verified that the temperature readings were displayed correctly on the LCD screen in real-time and cross checked voltages with DMM as well. Overall, this testing method allowed us to verify the accuracy and reliability of the temperature readings displayed on the LCD screen.

4.3 LCD Display (TFT LCD Screen)

4.3.1 Testing Setup & Method

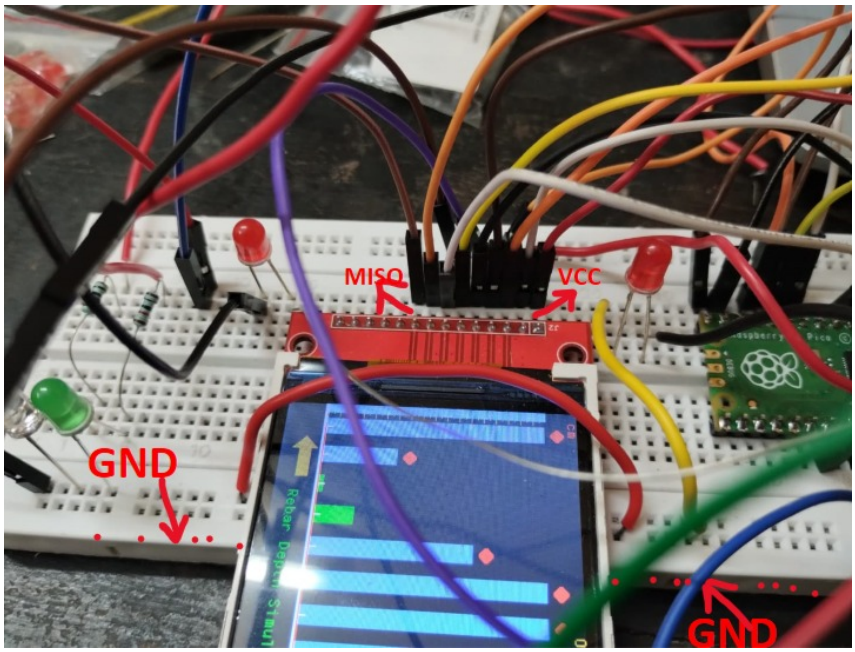


Figure 11: Sensor setup

The display has two modes: SPI and 8-bit. We used the SPI mode with nine pins, whereas the 8-bit mode uses twelve for interfacing. The pin connections are as shown in the schematic in the beginning. After connecting the LCD, we used the online library code for driving the ILI9341 driver. Functions available in the library were used for tasks such as initializing the LCD and sending pixel data and commands via the microcontroller.
We made specific fundamental changes in the driver code, such as changing SPI Port no. to GPIO 1, MOSI pin to GPIO 11, MISO pin to GPIO 12, SCK pin to GPIO 10, CS pin to GPIO 13, DC pin to GPIO 14 and RST pin to GPIO 15 pins and connected LCD pins according to the schematic.

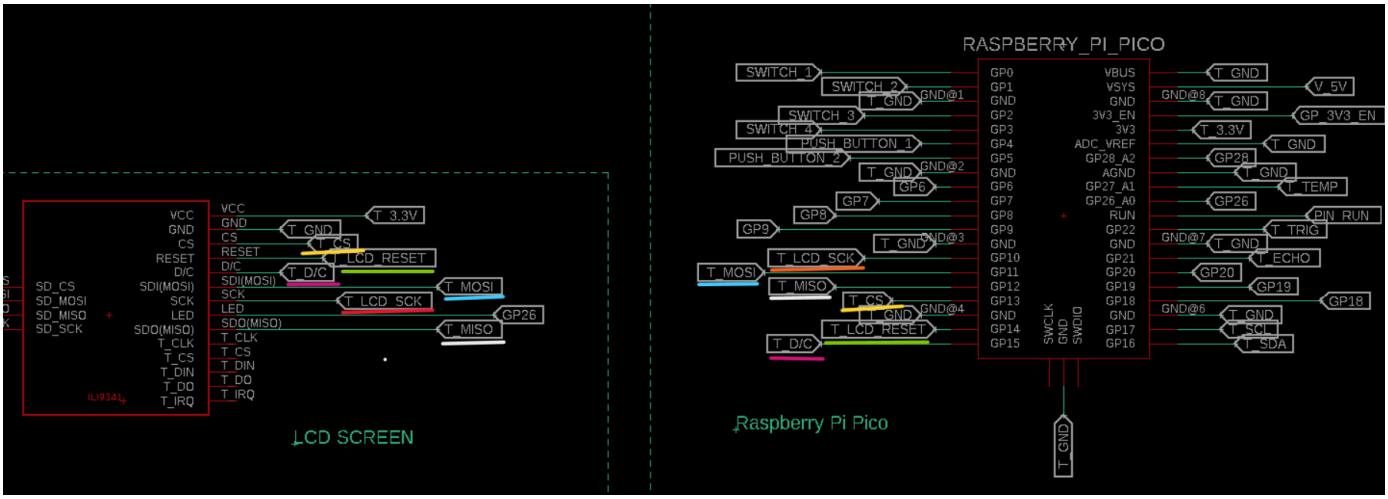


Figure 12: LCD setup

Testing Method

We used the functions from the library for driving ILI9341 and ran the demo code given along

with it, which is supposed to print a sentence in different text and background colours on the screen.

4.3.2 Testing Results

The text was displayed as expected on the LCD screen, and we can now use this base code to control individual pixels and paint the User Interface on the screen according to our design. The revised User Interface designs are as follows :

4.4 Ultrasonic Distance Sensor (HC-SR04)

Overview
This sensor uses ultrasonic waves to measure the distance to an object. It sends a high-frequency sound wave to calculate distance, and the sensor has a range of 4m with an accuracy of 3mm. As explained in the User Interface section, we used the distance value obtained to simulate iPEC probe functionality and also to diplay the distance from wall.

4.4.1 Testing Setup & Method

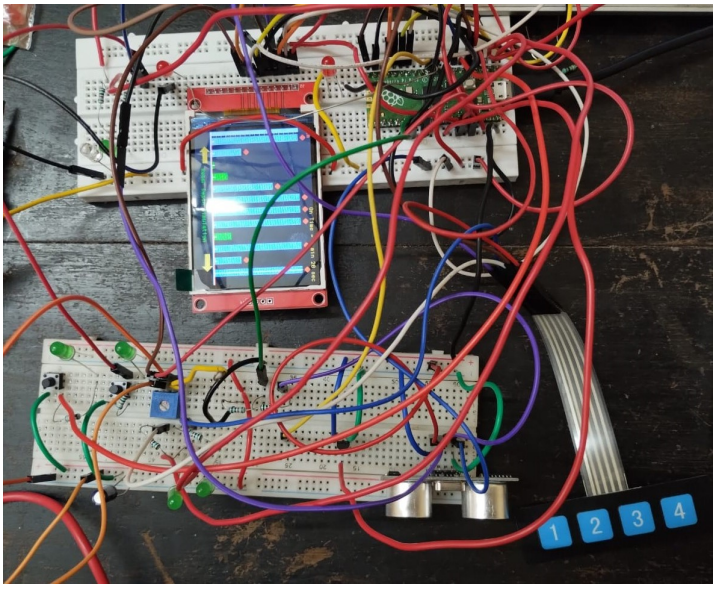


Figure 13: Sensor Setup and whole view

- **Hardware Setup**: This sensor has four pins; they are V_{CC}, GND, TRIG, and ECHO, which are connected according to the schematic.V_{CC} will be connected to 5V on board, and GND is connected to the ground. The TRIG pin is connected to GPIO 22, and the ECHO pin is connected to GPIO 21.

  As Raspberry Pi Pico GPIO pins i/o voltage is 3.3V, we used two 10K resistors in series from ECHO to GND; we use these as a divider to convert the 5V logic level to a safe 2.5V. For Raspberry Pi Pico, any voltage between 1.8V and 3.3V will be read by the Raspberry Pi as high, and anything lower than 1.8V is read as low.

- **Software Setup**: ECHO GPIO 21 pin and TRIG GPIO 21 pin are initialized as GPIO pins, and their directions are set as INPUT and OUTPUT, respectively, using the gpio_set_dir() function. The microcontroller calculates the speed using the time the sound wave takes, which is measured using an internal clock.

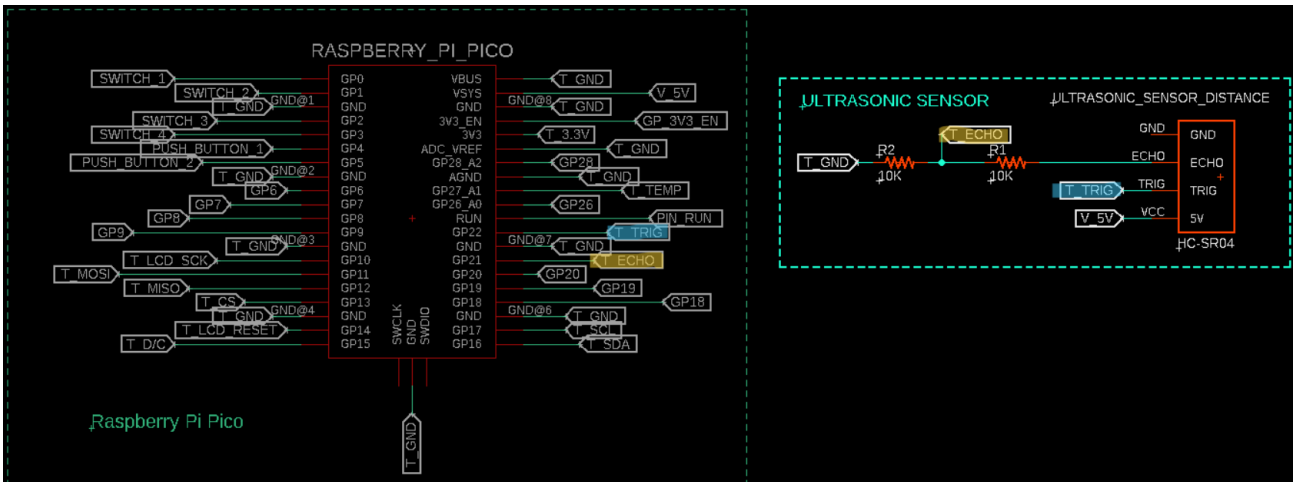


Figure 14: Sensor setup

Testing Method

The basic principle of operation for ultrasonic distance sensor:

1. Using IO trigger for at least 10us high-level signal
2. The Module automatically sends eight 40 kHz and detects whether there is a pulse signal back.
3. IF the signal is back, through a high level, time of high output IO duration is the time from sending ultrasonic to returning.
4. It measures the distance by using the following formula

$$\text{distance} = \text{speed} * \text{time}/2$$

Implementation:

- We set the TRIG pin(GPIO 22) high for 20us. Just after that, as soon as the ECHO pin(GPIO 21) goes high, we initialize a variable named start_time to save time at that instant.
- After that, we run a while loop and wait till the ECHO pin goes low, and thus subtract from the current time from start_time to obtain the time for which the ECHO pin was high.
- We then multiply the time taken by speed calculated using room temperature obtained from the temperature sensor and divide by 2 to get the distance from the object in front of the sensor.
- We placed an object(book) in front of the sensor and a ruler alongside and checked the distance displayed on LCD and that seen on the ruler for testing.

4.4.2 Testing Results

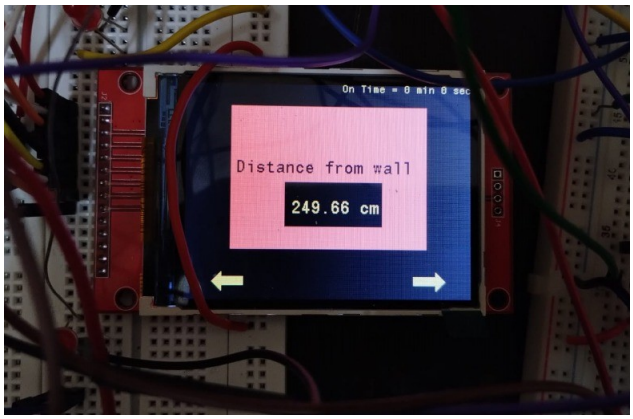


Figure 15: Result

- We found that the distance measured by the sensor at various lengths ranging from 3cm to 300+ m was entirely accurate.
- **Integration Testing**:Initially, we used a fixed value of sound 346 m/s for distance calculation and later integrated temperature sensor readings for better accuracy. We later integrated the ultrasonic distance sensor subsystem with the temperature sensor subsystem and displayed the values on the LCD screen in real time.
- Thus, we verified the accuracy and reliability of the distance readings by displaying them on the LCD screen and comparing them with measured distances along the scale.

4.5 Potentiometer

Overview

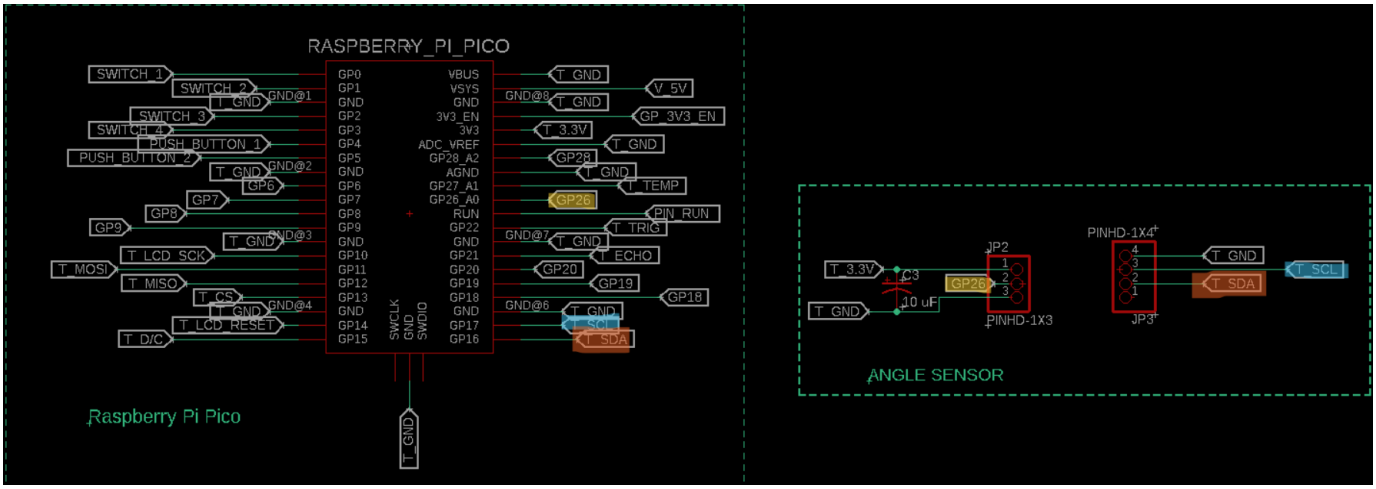


Figure 16: Sensor setup

We used a potentiometer to adjust the brightness of the LCD screen. The sensor will read the analog value, and the brightness will be set accordingly. Since we didn't test the angle sensor, we programmed its UI, which displays gradually on the screen as we increase brightness using a potentiometer. We rotate the potentiometer and, depending on that, the brightness is set.

4.5.1 Testing Setup & Results

- **Hardware Setup**: We are using potentiometer output as input to microcontroller ADC to adjust brightness. It is connected on one side to 3.3V and 0V on the other.

  The pot output is connected to LED pin of LCD screen which adjusts its brightness.

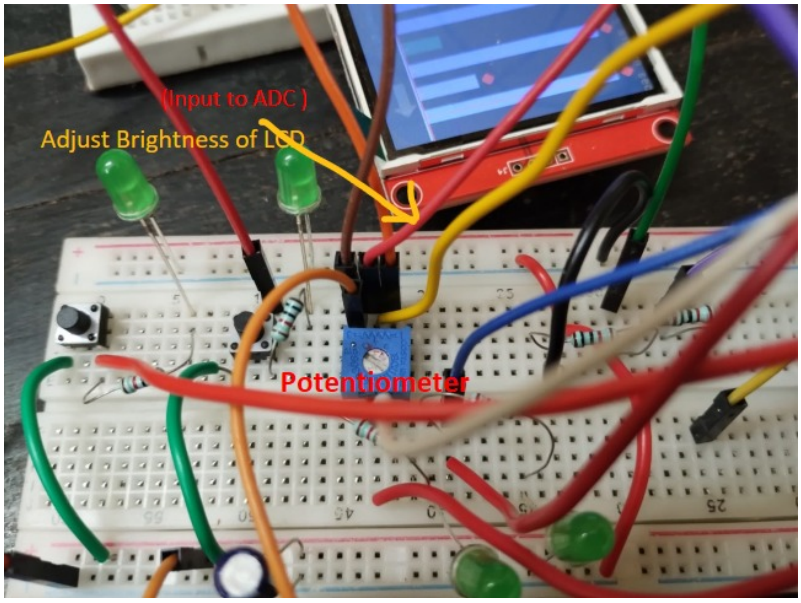


Figure 17: Setup

- **Software Setup**: We connect the potentiometer output to GPIO 28 which is ADC2 in our board. Similar to temperature sensor we read the voltage values from the ADC.

Testing Method

- We turn the potentiometer screw to get voltage between 0 and 3.3V which is sent to GPIO28 as well as LED pin of LCD screeen.
- From GPIO28 we read the voltage value using ADC, scale it appropriately and display the brightness level on LCD in UI by showing horizontal bars which get gradually filled/emptied as we increase/decrease brightness. Each bar corresponds to 25% brightness (= total 4 bars).

Figure 18: Result

4.6 Buttons and LEDs

Overview
We used a membrane keypad and push buttons as switches for taking input from the user. They are interfaced with the microcontroller through GPIO pins that detect the switches' state and take appropriate action based on it. The buttons with their functions and types are as follows :
Membrane Keypad buttons
1. Left (1) [to go to previous screen]
2. Right (2) [to got to next screen]
3. Start (3) [to turn the device on/off]
4. Home/Esc (4) [to return to Home Screen/ exit popup window]

Push buttons
5. Info button [display popup window with information]
6. OK button [pause/resume rebar simulation]

4.6.1 Testing Setup & Method

- **Hardware Setup**: We used pull-down switches to take input from the user. These switches are connected to GPIO pins, as shown in the schematic. They are also connected to respective LEDs with a series resistor that glows when we press separate buttons.

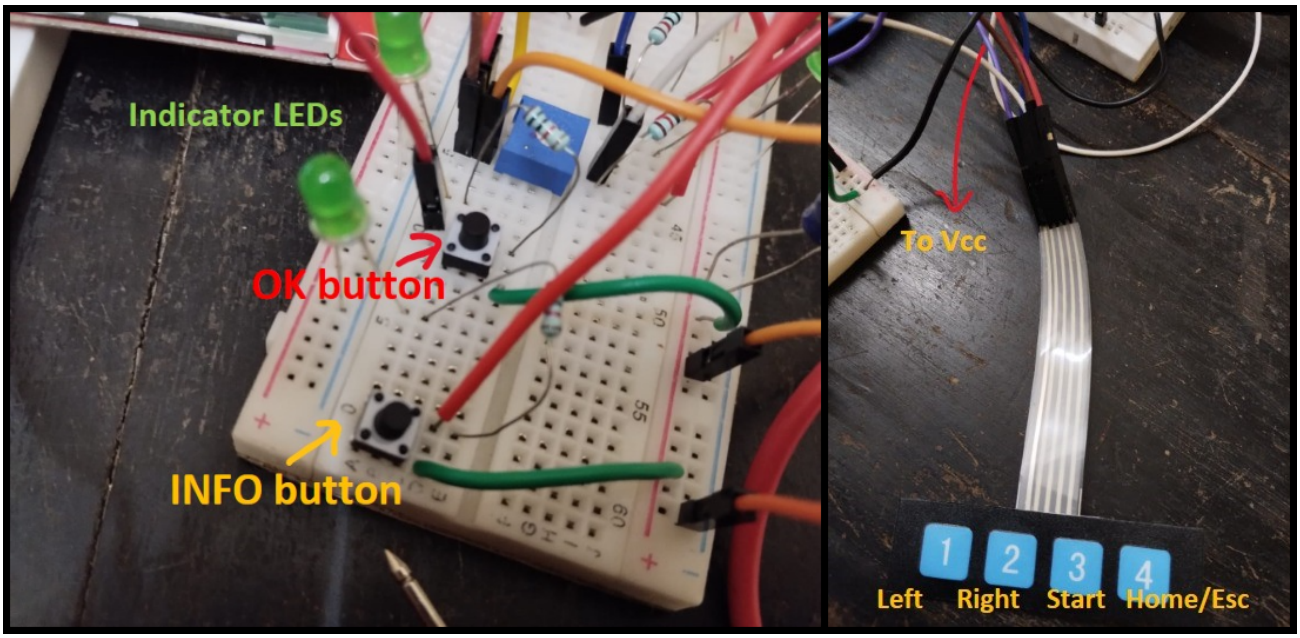


Figure 19: Hardware setup

- **Software Setup**: The GPIO pins corresponding to the buttons are initialized as GPIO pins, and the direction is set to INPUT using gpio_set_dir() and set to pull-down mode by gpio_pull_down() function. We also implemented Key Debouncing using software method.

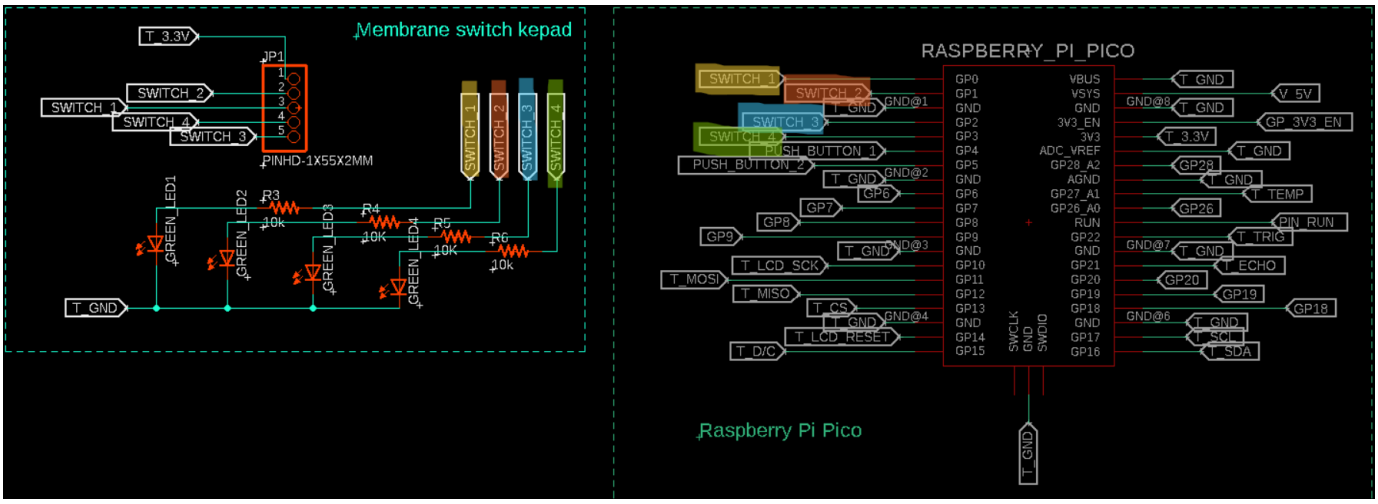


Figure 20: Buttons setup

Testing Method

- All buttons are connected to LEDs to check whether they got pressed. Resistors are connected in series with LEDs to limit current flow and prevent LEDs from damage.
- A continuous while loop that keeps running checks at the beginning of any key press, and a specific action happens depending on the button pressed and the current UI Screen. We kept one more LED that glows after the program finds the button press. This allowed us to see the delay between the time button is pressed and when the microcontroller acknowledges it and calls the required functions.
- Thus, the buttons and membrane keypad were directly tested with the entire system by printing on an LCD screen to check their functioning.

- We often checked that the user interface gets affected as desired by pressing buttons and seeing the output on the LCD screen.

4.6.2 Testing Results

- The push buttons had a relatively small delay. Both the LEDs glowed almost simultaneously, indicating the signal reached the within significantly less time to the microcontroller and was acknowledged as well.
- The membrane-keypad buttons sometimes had a significant delay (2-3 seconds) and had to be pressed for that long for the board to take action. However, most of the time, the delay was within 1 second.
- We must use a new membrane keypad or change our program to reduce this delay and ensure a smooth experience. We will implement a software solution called Key Debouncing which may solve this issue.

5 Programming Logic Overview

Firstly we initialize all the GPIO pins corresponding to buttons, LEDs, sensors, etc., all sensors, ADCs on the board and most importantly, the LCD screen using functions. After that, a while loop is run, continuously checking various values depending on the current UI screen and other conditions. We maintain an array _framebuffer to store the data of all pixels, which is sent to the LCD screen whenever we wish to display it. This is called Rendering, in which the UI is updated on the LCD screen. Inside the while loop, we first check if any keys/buttons are pressed and if required actions are taken as needed. Later instead of this, we switched to interrupts for less delay and smoother experience. After that, we check whether to Render the screen or not. Initially, we rendered every time we entered the while loop. However, since rendering/ updating LCD screen doesn't' take little time, it affected the button inputs drastically. Hence, to reduce such delays and unwanted freezing of the screen, which occurred originally, we render it a moderate rate of 30 FPS. At this rate, there weren't any such significant issues.
Depending on the current screen, respective data is displayed on the LCD screen. And the next time we again enter the Main Loop, depending on the button pressed, the _framebuffer is updated. Later, while rendering, this change is communicated to the LCD screen and is visible to us. We can manipulate the UI whichever we want and display relevant information on the screen based on how we handle vital presses. Also, since LCD screen uses RGB565, we have to convert our RGB pixel values to that format. For displaying icons such as thermometer/cross, we converted png images to C arrays and used those for printing.

6 CAD Design

The CAD layout for our system is designed to package the PCB in a compact and ergonomic enclosure. The enclosure features a rectangular hole on the top edge to make the LCD visible. A piece of the top face is cut to position the membrane buttons used for user input. Two small circular holes are made to fit the dome buttons that are used to display the menu and information. These buttons are attached to LEDs to provide feedback to the user, and small holes are made to allow the LED to glow when the corresponding button is pressed.
On the back view of the enclosure, a rectangular hole is made to accommodate the ultrasonic distance sensor. The sensor is designed to pop out slightly from the enclosure, requiring unobstructed access to send and receive ultrasonic waves. The enclosure is designed to be sturdy and compact, with a minimalistic and user-friendly design that allows for easy access to all the

buttons and sensors. Overall, the CAD layout provides a functional and aesthetically pleasing enclosure for our HMI system, which is ergonomic and intuitive.

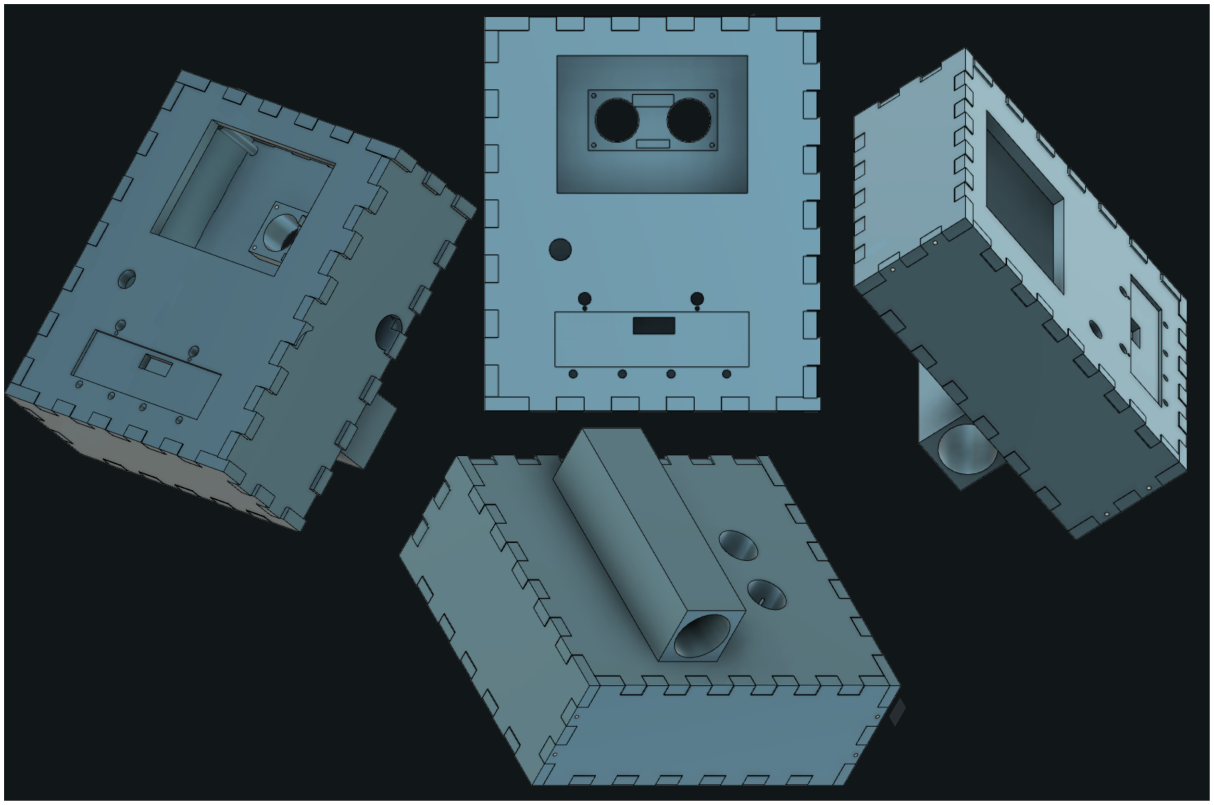


Figure 21: CAD design

7 Challenges Faced

- Delay in action after key press
  - Used interrupts to check key presses in a while loop
- Slow rendering of UI on LCD
  - Instead of updating LCD screen every time we enter while loop, we update it at regular intervals (30 FPS)
- Making the design ergonomic and user-friendly
  - Rearranged components and PCB connections to make it more compact and easy to handle

8 Conclusion and Future Work

8.1 Conclusion

- Implemented HMI system with user-friendly interface with features such as navigation, pausing simulation, and popup screens.
- Implemented user-interface in a way to mimic a rebar scanning device
- Designed and packaged the device to allow easy handling

- Further improvements may include making the design more compact and adding more features such as showing 2D mapping data

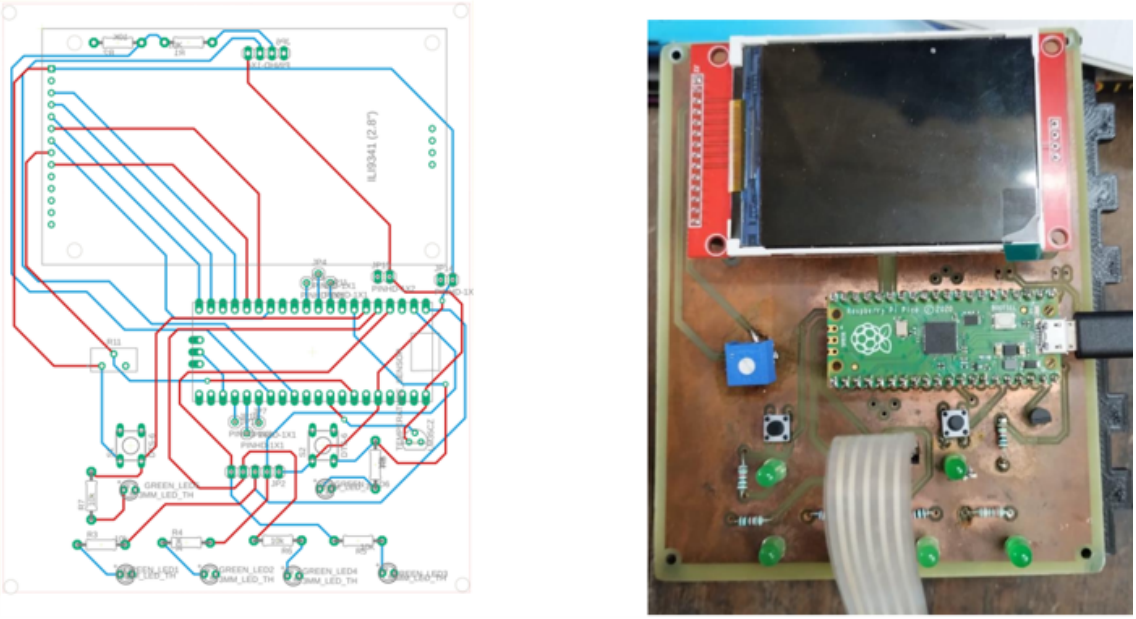


Figure 22: Final PCB

Figure 23: Final Product

8.2 Future Work

- **Data Integration**: Integrate the sensor data from the HMI device with the 2D mapping data subproject and the Annotation part to provide a visual representation and add context to the data.
- **Communication Protocols**: Explore protocols compatible with the other subprojects and ensure reliable communication in harsh environments. For example, the 2D mapping data subproject may require a specific protocol

- **Power Management**: Design the HMI device to conserve power or recharge it using the iPEC project's power source if necessary.
- **Data Analysis and Visualization**: Depending on the goals of the iPEC project, we need to explore ways to analyze and visualize the data collected by the HMI device and other subprojects to identify trends and patterns.
- **Environmental Sensors**: Consider adding environmental sensors to measure temperature, humidity, and light levels to provide a more complete picture of the environmental conditions.
- **Rugged Design**: Design the HMI device with waterproof and dustproof enclosures, high-contrast LCD screens, and materials that can withstand impact and vibration.
- **Field Calibration**: Design the device to support field calibration to ensure accurate measurements under harsh environmental conditions.
- **Real-time Data Analysis**: Design the device to perform real-time data analysis on the sensor data and provide alerts or notifications when certain conditions are met.

By integrating the HMI device with other subparts of the iPEC project, we can ensure that the project is successful and provides accurate and meaningful data even in harsh environmental conditions.

9 Links to Code and Video Demo

- Link to project code
- Link to breadboard circuit demo
- Link to working of the device demo
- Link to the folder containing CAD files:
- Link to the folder containing PCB files: