SRM Institute of Science and Technology

College of Engineering and Technology

Department of Electronics and Communication Engineering

18ECO109J-Embedded System Design Using Raspberry Pi
2022-23 (Even Semester)

Mini Project Report

Name : **RA2011003011105**
Register No. : **ABHIRUP MANDAL**

Day / Session : **1**

Venue : **TP 1317**

Project Title : **SMART HEALTHCARE SENSOR**

Lab Supervisor : **SUGANTHI BRINDHA G**

Team Members : **1) ASHUTOSH SHARMA (RA2011003011084)**
2) ABHIRUP MANDAL (RA2011003011105)
3) DEBARGHYA BARIK (RA2011026010022)

Particulars	**Max. Marks**	**Marks Obtained**
Objective & Description	05	
Algorithm,Flowchart,Program	20	
Demo verification	10	
Viva	10	
Report	05	
Total	**50**	

REPORT VERIFICATION

Date : **11/05/2023**

Staff Name : **SUGANTHI BRINDHA G**

SMART HEALTHCARE SENSOR

OBJECTIVE:

To collect, store and display live biomedical data raspberry pi pico.

ABSTRACT:

The Smart Healthcare System is a project designed to improve patient monitoring and healthcare management using Raspberry pi pico and AHD10 Sensor that has a 8232 ECG Sensor, and a Temperature and Humidity sensor.

The system aims to enhance healthcare outcomes by providing real-time health monitoring and effective management of patient's medical data.

The hardware component of the system includes the Raspberry Pi Pico microcontroller, the AHD10 Sensor, and various other sensors that collect vital health data from the patient.

The system provides several benefits to patients, doctors, and healthcare providers. Patients can receive personalized and timely care, as their health data is constantly monitored and analyzed.

Doctors and healthcare providers can access the patient's data from anywhere and at any time, allowing for remote consultations and timely interventions.

Additionally, the system can reduce the burden on healthcare facilities, as patients can be monitored remotely, reducing the need for hospitalization.

HARDWARE / SOFTWARE REQUIRED:

- Raspberry Pi Pico
- M/M, F/F and M/F jumper wires
- Breadboard
- ThingSpeak
- Firebase
- AHT10 Digital temperature and humidity sensor
- AD8232 ECG sensor

ALGORITHM:

TEMPERATURE AND HUMIDITY SENSOR:

1. Import necessary modules:

    a. machine - for controlling hardware devices

    b. network - for connecting to Wi-Fi

    c. urequests - for sending HTTP requests

    d. utime - for timing operations

2. Set up Wi-Fi credentials and ThingSpeak Write API key.

3. Configure I2C pins for humidity and temperature sensor (AHT10).

4. Set AHT10 sensor configuration commands:

    a. AHT10_ADDRESS - 7-bit address of the AHT10 sensor

    b. AHT10_INIT_CMD - initialization command to set up the AHT10 sensor

    c. AHT10_MEASURE_CMD - measurement command to read humidity and temperature data from the AHT10 sensor

5. Connect to Wi-Fi network using WLAN module.

6. Initialize the AHT10 sensor by writing the initialization command to the sensor.

7. Define a function to read the data from the AHT10 sensor.

8. Define a function to parse the raw data read from the AHT10 sensor into temperature and humidity values.

9. Set the total number of iterations to 30 (10 minutes) and loop through them.

10. Read the data from the AHT10 sensor.

11. Parse the raw data read from the AHT10 sensor into temperature and humidity values.

12. Send the data to ThingSpeak using a POST request:

    a. Construct the payload string containing the temperature and humidity data.

    b. Construct the URL for the ThingSpeak API update endpoint, including the Write API key and payload.

    c. Set a maximum number of retries and retry interval.

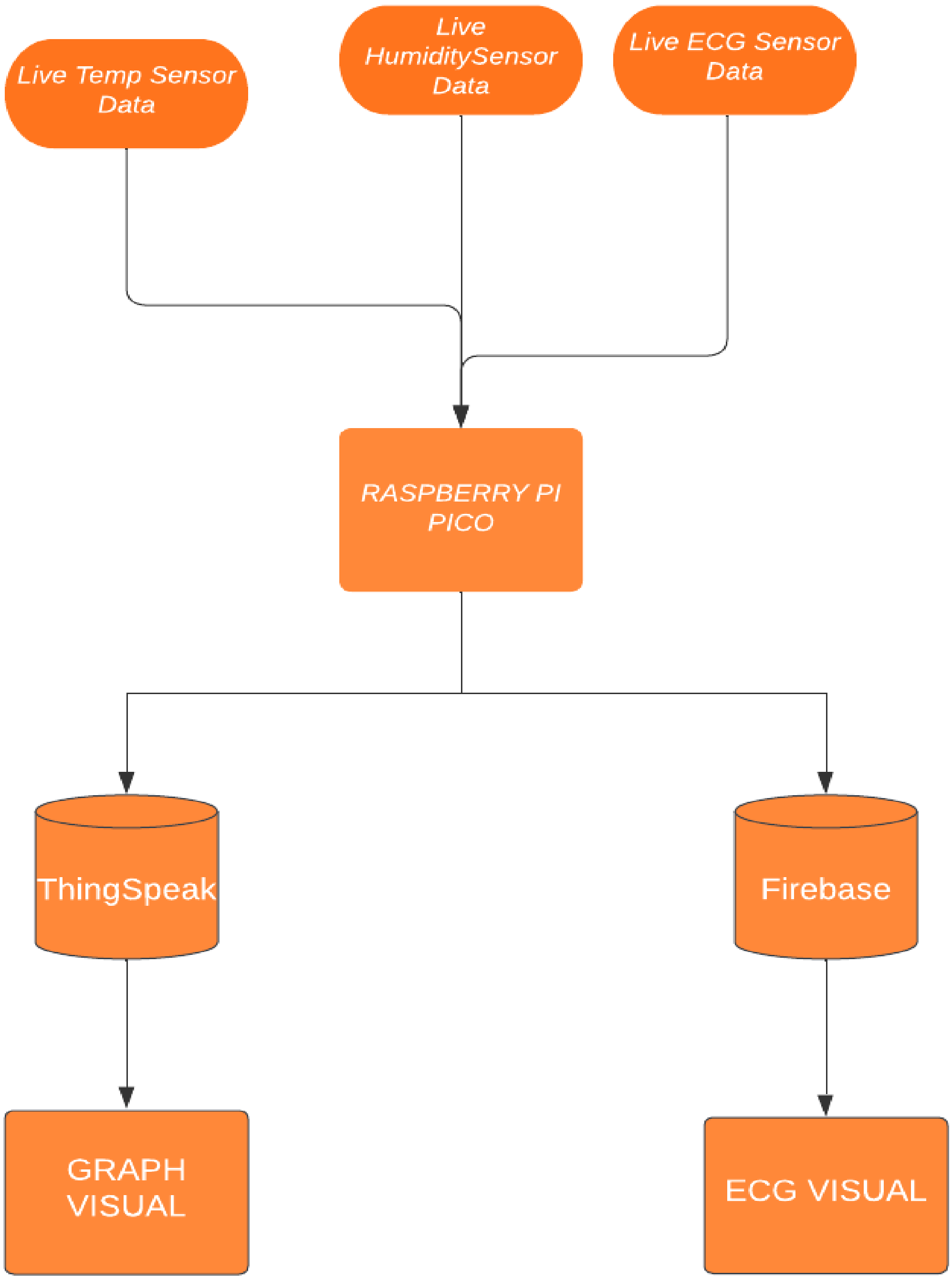
    d. Attempt to send the data to ThingSpeak using the urequests.post() function.

13. End of program.

ECG SENSOR:

1. Import necessary libraries - machine, utime, and urequests.
2. Configure pins for the ECG sensor - the lo_minus_pin, lo_plus_pin, and output_pin.
3. Set up the Firebase URL and authentication key for data logging.
4. Define a function leads_connected() to check if the ECG leads are connected properly.
5. Define a function log_data(sample_interval) to continuously log ECG data to Firebase. The function takes a sample_interval argument to specify the time interval between each sample.
6. Inside the log_data() function, use a while loop to continuously check for the ECG leads connection status.
7. If the leads are connected, read the ECG value from the output_pin and create a data dictionary with the timestamp and ECG value.
8. Use urequests library to send a POST request to the Firebase URL with the data dictionary and authentication headers.
9. Check if the response status code is 200, which indicates a successful data upload to Firebase. If successful, print a message with the timestamp and ECG value. If not, print an error message with the response text.
10. Close the response object and wait for the specified sample_interval before taking the next ECG sample.
11. If the leads are disconnected, print an error message and wait for 1 second before checking the connection status again.
12. Call the log_data() function with the sample_interval parameter set to 100 ms to log ECG data indefinitely.
13. The program will continuously log ECG data to Firebase until it is interrupted.

FLOW CHART:


Live Temp Sensor Data
Live HumiditySensor Data
Live ECG Sensor Data
RASPBERRY PI PICO
ThingSpeak
Firebase
GRAPH VISUAL
ECG VISUAL

PROGRAM:

TEMPERATURE AND HUMIDITY SENSOR:

```
import machine
import network
import urequests
import utime

# Set up your Wi-Fi credentials
WIFI_SSID = "WIFI_SSID"
WIFI_PASSWORD = "WIFI_PASSWORD"

# Set up your ThingSpeak Write API Key
WRITE_API_KEY = "API_KEY"

# I2C configuration
sda_pin = machine.Pin(4)  # SDA pin (GP0)
scl_pin = machine.Pin(5)  # SCL pin (GP1)
i2c = machine.SoftI2C(sda=sda_pin, scl=scl_pin, freq=400000)

AHT10_ADDRESS = 0x38
AHT10_INIT_CMD = bytearray([0xE1, 0x28, 0x00])
AHT10_MEASURE_CMD = bytearray([0xAC, 0x33, 0x00])

# Connect to Wi-Fi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_SSID, WIFI_PASSWORD)
print("Connecting to Wi-Fi...")  # Debugging print statement
```

```
    while not wlan.isconnected():
        pass
    print("Connected to Wi-Fi!")  # Debugging print statement

    # Set the total number of iterations to 300 (10 minutes)
    num_iterations = 30

    def initialize_sensor():
        i2c.writeto(AHT10_ADDRESS, AHT10_INIT_CMD)
        utime.sleep_ms(100)

    def read_data():
        i2c.writeto(AHT10_ADDRESS, AHT10_MEASURE_CMD)
        utime.sleep_ms(100)
        data = i2c.readfrom(AHT10_ADDRESS, 6)
        return data

    def parse_data(data):
        humidity_raw = ((data[1] << 12) | (data[2] << 4) | (data[3] >> 4)) / 0x100000
* 100
        temperature_raw = (((data[3] & 0x0F) << 16) | (data[4] << 8) | data[5]) /
0x100000 * 200 - 50
        return temperature_raw, humidity_raw

    initialize_sensor()

    for i in range(num_iterations):
        raw_data = read_data()
        humidity, temperature = parse_data(raw_data)
```

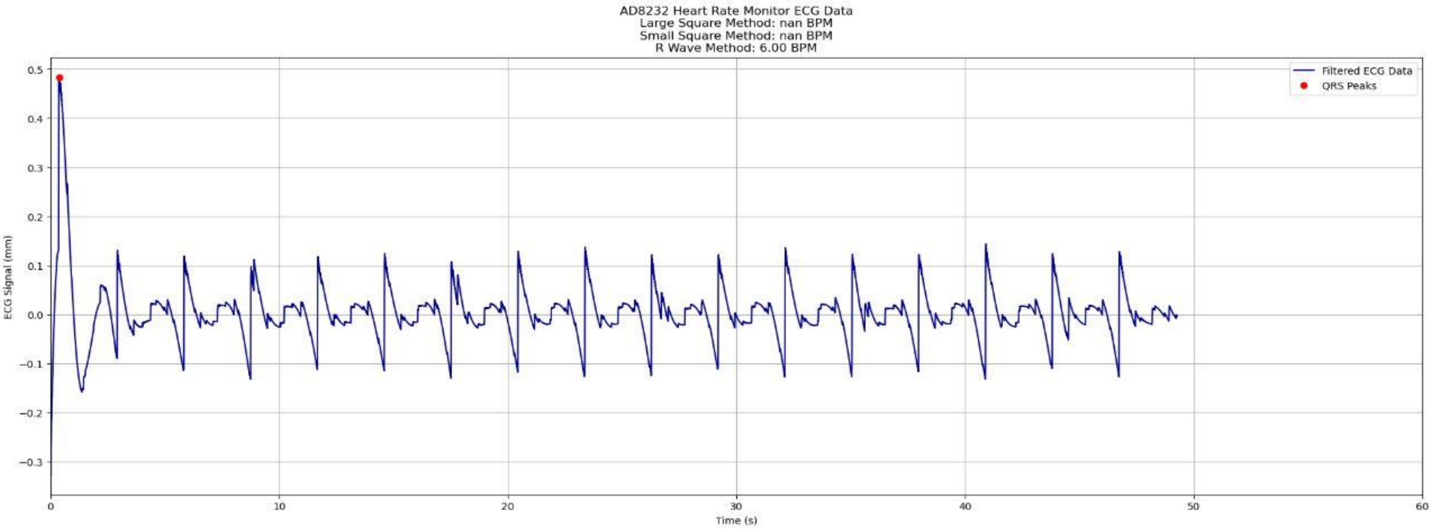
```
    # Send the data to ThingSpeak
    payload = "field1={:.2f}&field2={:.2f}".format(temperature, humidity)
    url =
"https://api.thingspeak.com/update?api_key={}".format(WRITE_API_KEY)

    # Retry mechanism
    max_retries = 5
    retry_interval = 5  # in seconds
    retries = 0

    while retries < max_retries:
        try:
            response = urequests.post(url, data=payload)
            print("Temperature: {:.2f} °C, Humidity: {:.2f} %, Response:
{}".format(humidity, temperature, response.text))
            break
        except OSError as e:
            print("Error occurred: {}".format(e))
            print("Retrying in {} seconds...".format(retry_interval))
            utime.sleep(retry_interval)
            retries += 1

    utime.sleep(2)
```

ECG SENSOR:

```
ECG Sensor:import machine
import utime
import urequests

# Configure pins
lo_minus_pin = machine.Pin(2, machine.Pin.IN)
lo_plus_pin = machine.Pin(3, machine.Pin.IN)
output_pin = machine.ADC(machine.Pin(26))

# Configure Firebase
firebase_url = 'https://pi-pico-25296-default-rtdb.asia-
southeast1.firebasedatabase.app/ecg_data.json'
firebase_auth = 'auth_key'

# Check if the leads are connected properly
def leads_connected():
    if lo_minus_pin.value() == 0 and lo_plus_pin.value() == 0:
        print("Lead Connected")

    return lo_minus_pin.value() == 0 and lo_plus_pin.value() == 0

# Log ECG data to Firebase
def log_data(sample_interval):
    while True:
        if leads_connected():
            timestamp = utime.ticks_ms()
            ecg_value = output_pin.read_u16()
            data = {
```

```
                "timestamp": timestamp,
                "ecg_value": ecg_value
            }
            headers = {'Content-Type': 'application/json'}
            response = urequests.post(firebase_url + '?auth=' + firebase_auth,
json=data, headers=headers)
            if response.status_code == 200:
                print(f"Data successfully sent to Firebase: {timestamp}, {ecg_value}")
            else:
                print(f"Failed to send data to Firebase: {response.text}")
            response.close()
            utime.sleep_ms(sample_interval)
        else:
            print("Leads disconnected. Please check the connections.")
            utime.sleep_ms(1000)

# Log data indefinitely with a sample interval of 100 ms
log_data(100)
```

OUTPUT:


AD8232 Heart Rate Monitor ECG Data
Large Square Method: nan BPM
Small Square Method: nan BPM
R Wave Method: 6.00 BPM
Filtered ECG Data
QRS Peaks
ECG Signal (mm)
Time (s)

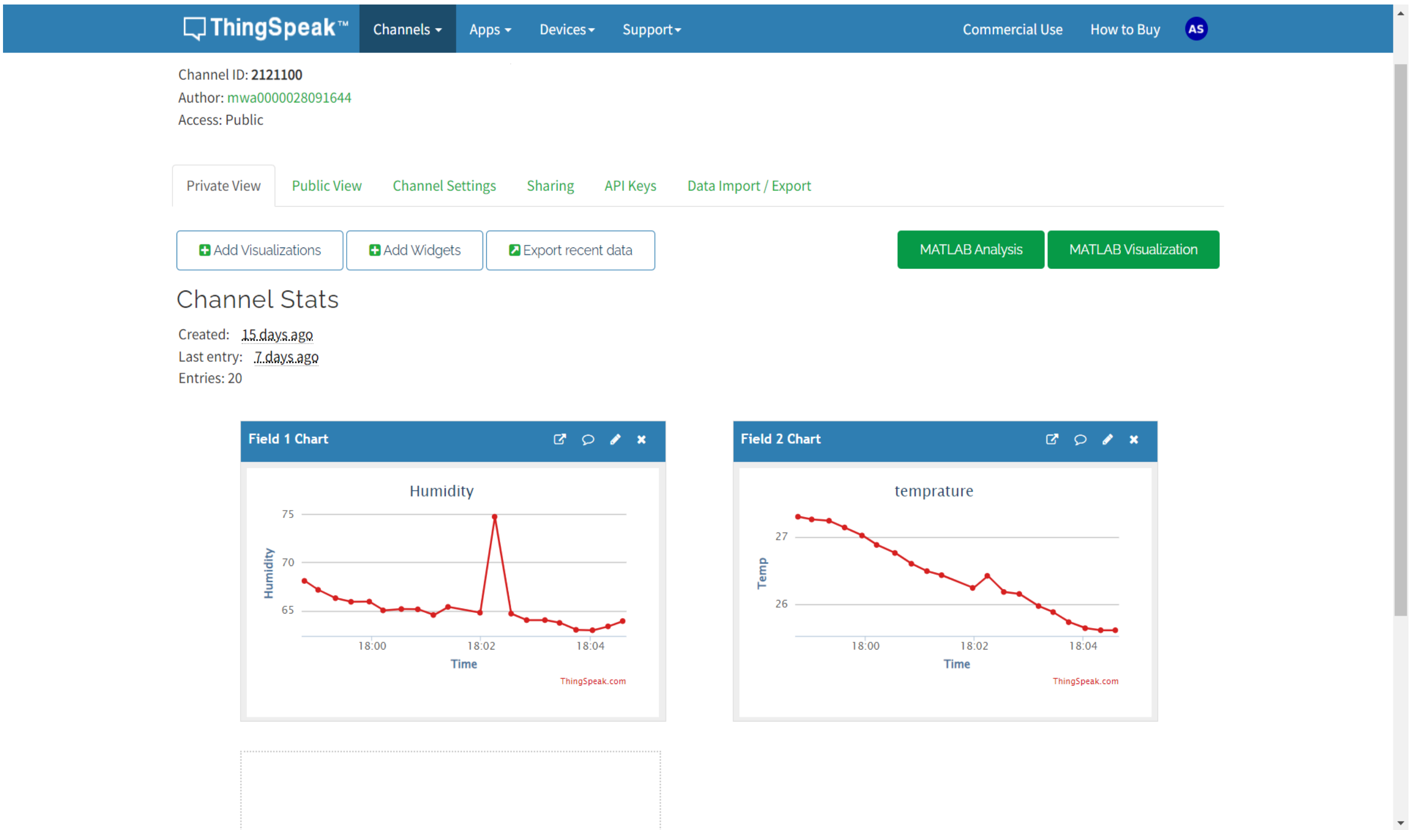
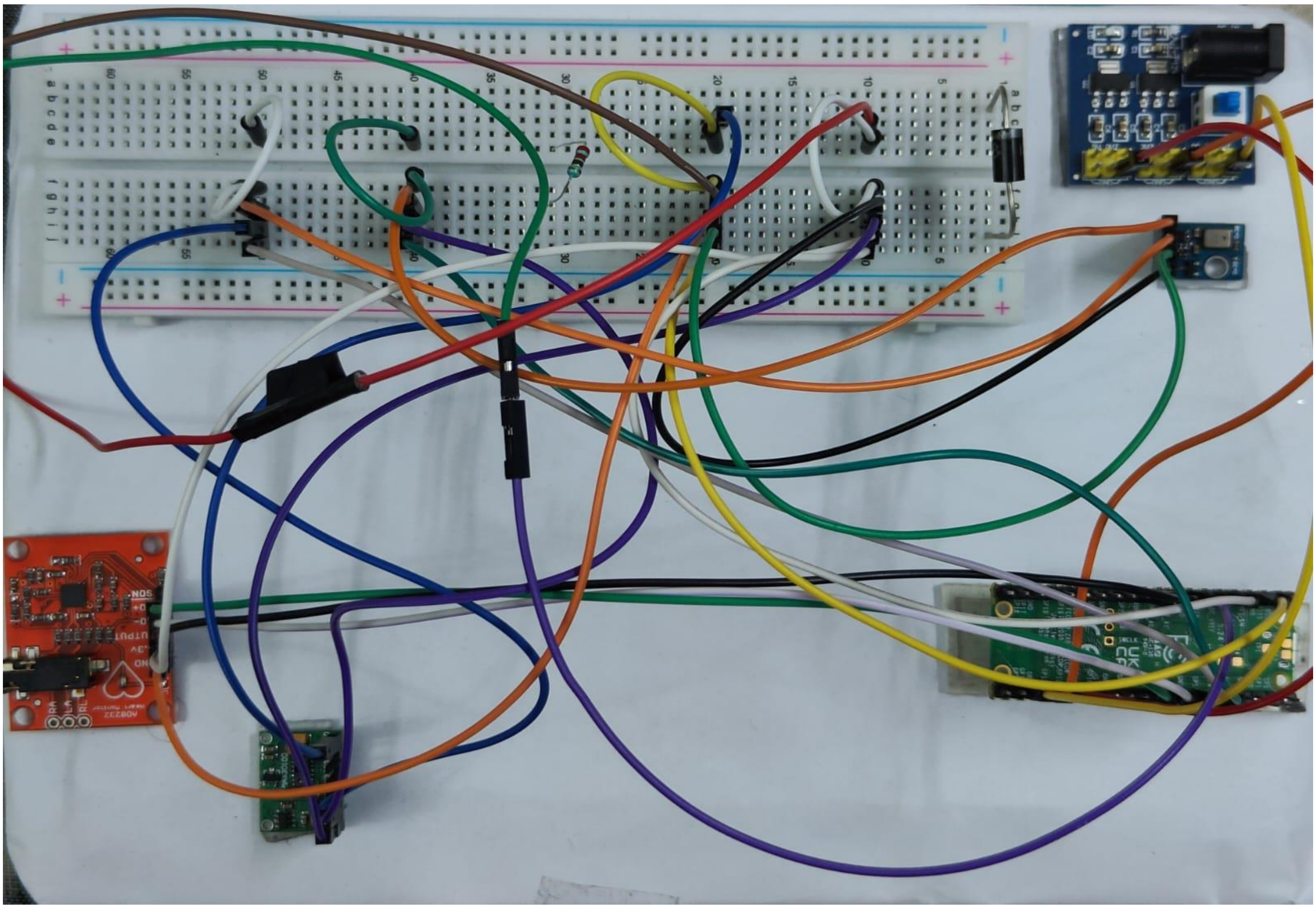
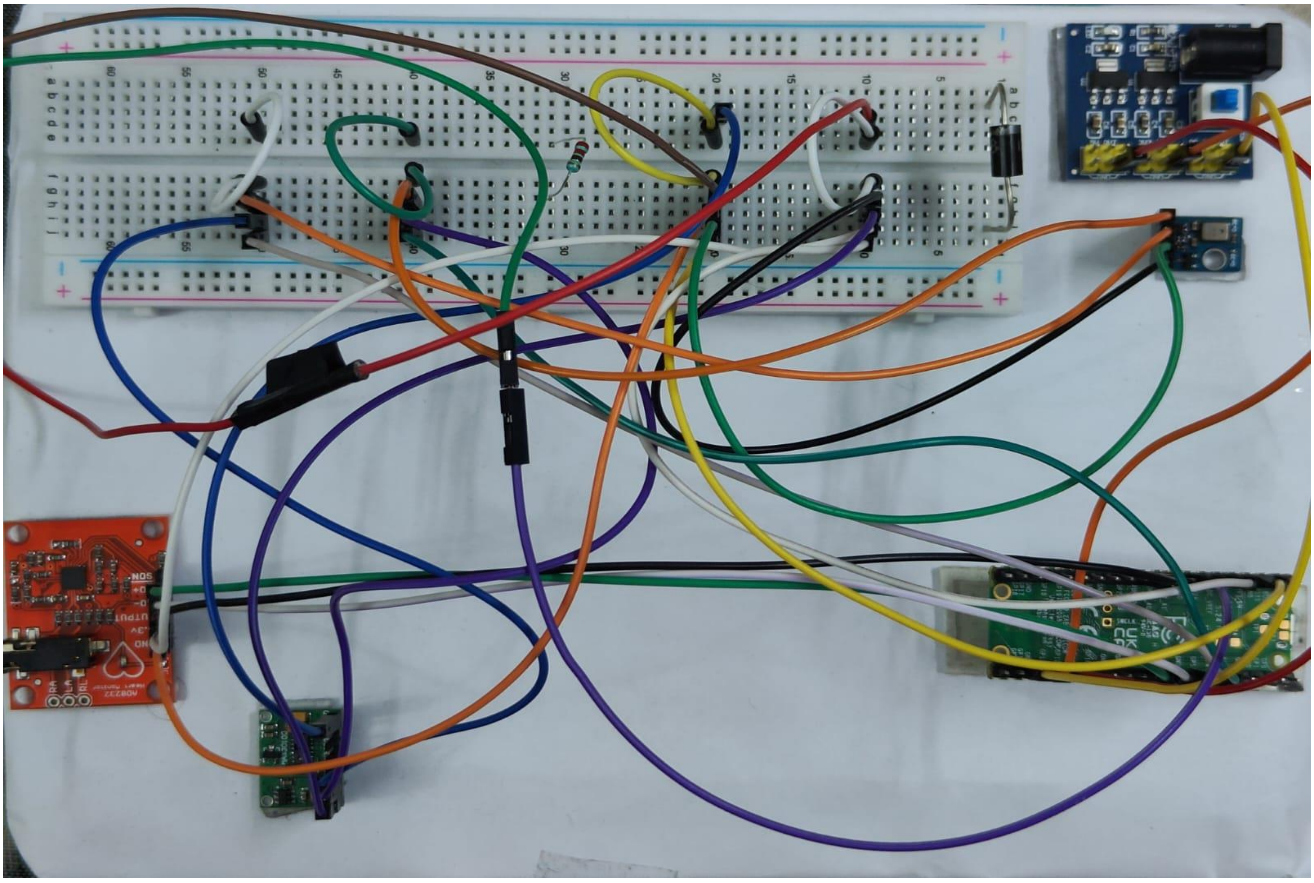

ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Channel ID: 2121100
Author: mwa0000028091644
Access: Public
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Add Visualizations
Add Widgets
Export recent data
MATLAB Analysis
MATLAB Visualization
Channel Stats
Created: 15 days ago
Last entry: 7 days ago
Entries: 20
Field 1 Chart
Humidity
Time
ThingSpeak.com
Field 2 Chart
temprature
Temp
Time
ThingSpeak.com

DIAGRAM:



IMPLEMENTATION:



VIDEO LINK:
https://drive.google.com/file/d/1BD4WyzYVoVM1ahyTLkDXLM61tA9Pj3Rs/view?usp=sharing

CONCLUSION:

In conclusion, the Smart Healthcare System utilizing Raspberry Pi Pico and various sensors, such as temperature, humidity, and ECG sensors, provides a powerful tool for monitoring user health data. With the ability to constantly track and upload data to online platforms, healthcare providers can easily access and analyze patient data to provide more personalized and effective care. Furthermore, this system has the potential to identify early warning signs of health issues, allowing for timely intervention and treatment. Overall, the Smart Healthcare System has the potential to significantly improve healthcare outcomes and enhance the quality of life for patients.

REFERENCES:

https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html
https://www.analog.com/media/en/technical-documentation/data-sheets
https://www.electroschematics.com/temperature-sensor/
https://lastminuteengineers.com/max30100-pulse-oximeter-heart-rate-sensor