The Kavery Engineering College

Mecheri, Salem Dt, Tamilnadu– 636 453.
(Approved by AICTE, New Delhi & Affiliated to Anna University)
An ISO 9001:2015 certified Institution and accredited by NAAC with A+ grade.

DEPARTMENT

OF

ELECTRICAL AND ELECTRONICS ENGINEERING

LABORATORY MANUAL

OCS352 IOT CONCEPTS AND APPLICATIONS LABORATORY

NAME OF THE STUDENT	
REGISTER NUMBER	
YEAR	
SEMESTER	

Prepared By
SENTHILKUMAR K
AP/EEE
THE KAVERY ENGINEERING COLLEGE

OCS352 **IOT CONCEPTS AND APPLICATIONS** **L T P C 2 0 2 3**

OBJECTIVES:

- To apprise students with basic knowledge of IoT that paves a platform to understandphysical and logical design of IOT
- To teach a student how to analyse requirements of various communication models andprotocols for cost-effective design of IoT applications on different IoT platforms.
- To introduce the technologies behind Internet of Things(IoT).
- To explain the students how to code for an IoT application using Arduino/Raspberry Piopen platform.
- To apply the concept of Internet of Things in real world scenario.

UNIT I INTRODUCTION TO INTERNET OF THINGS **5**

Evolution of Internet of Things – Enabling Technologies – IoT Architectures: oneM2M, IoT World Forum (IoTWF) and Alternative IoT Models – Simplified IoT Architecture and Core IoT Functional Stack – Fog, Edge and Cloud in IoT

UNIT II COMPONENTS IN INTERNET OF THINGS **5**

Functional Blocks of an IoT Ecosystem – Sensors, Actuators, and Smart Objects – Control Units - Communication modules (Bluetooth, Zigbee,Wifi, GPS, GSM Modules)

UNIT III PROTOCOLS AND TECHNOLOGIES BEHIND IOT **6**

IOT Protocols - IPv6, 6LoWPAN, MQTT, CoAP - RFID, Wireless Sensor Networks, BigData Analytics, Cloud Computing, Embedded Systems.

UNIT IV OPEN PLATFORMS AND PROGRAMMING **7**

IOT deployment for Raspberry Pi /Arduino platform-Architecture –Programming – Interfacing – Accessing GPIO Pins – Sending and Receiving Signals Using GPIO Pins – Connecting to the Cloud.

UNIT V IOT APPLICATIONS **7**

Business models for the internet of things, Smart city, Smart mobility and transport, Industrial IoT, Smart health, Environment monitoring and surveillance – Home Automation – Smart Agriculture

30 PERIODS

PRACTICAL EXERCISES: 30 PERIODS

1. Introduction to Arduino platform and programming
2. Interfacing Arduino to Zigbee module
3. Interfacing Arduino to GSM module
4. Interfacing Arduino to Bluetooth Module
5. Introduction to Raspberry PI platform and python programming
6. Interfacing sensors to Raspberry PI
7. Communicate between Arduino and Raspberry PI using any wireless medium
8. Setup a cloud platform to log the data
9. Log Data using Raspberry PI and upload to the cloud platform
10. Design an IOT based system

OUTCOMES:

CO 1: Explain the concept of IoT.
CO 2: Understand the communication models and various protocols for IoT.
CO 3: Design portable IoT using Arduino/Raspberry Pi /open platform
CO 4: Apply data analytics and use cloud offerings related to IoT.
CO 5: Analyze applications of IoT in real time scenario.

TOTAL PERIODS:60

TEXTBOOKS

1. Robert Barton, Patrick Grossetete, David Hanes, Jerome Henry, Gonzalo Salgueiro, “IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things”,CISCO Press, 2017
2. Samuel Greengard, The Internet of Things, The MIT Press, 2015

REFERENCES

1. Perry Lea, “Internet of things for architects”, Packt, 2018
2. Olivier Hersent, David Boswarthick, Omar Elloumi , “The Internet of Things – Keyapplications and Protocols”, Wiley, 2012
3. IOT (Internet of Things) Programming: A Simple and Fast Way of Learning, IOT KindleEdition.
4. Dieter Uckelmann, Mark Harrison, Michahelles, Florian (Eds), “Architecting the Internet ofThings”, Springer, 2011.
5. ArshdeepBahga, Vijay Madisetti, “Internet of Things – A hands-on approach”, UniversitiesPress, 2015
6. https://www.arduino.cc/

https://www.ibm.com/smarterplanet/us/en/?ca=v_smarterplanet

SUBJECT CODE: OCS352

SUBJECT NAME : IOT CONCEPTS AND APPLICATIONS

List of experiments

1. Write Basic and arithmetic Programs Using Embedded C.

2. Introduction to Arduino platform and programming

3. Explore different communication methods with IoT devices (Zigbee, GSM, Bluetooth)

4. Introduction to Raspberry PI platform and python programming

5. Interfacing sensors with Raspberry PI

6. Communicate between Arduino and Raspberry PI

7. Setup a cloud platform to log the data

8. Log Data using Raspberry PI and upload to the cloud platform

9. Design an IOT based system

S.NO	DATE	NAME OF THE EXPERIMENT	MARK	SIGNATURE

Write Basic and arithmetic Programs Using Embedded C.

AIM:

To write and execute Embedded Language Program for Addition, Subtraction, Multiplication, Modulation and Division of two 8-Bit numbers using Keil uVision5.

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	Keil uVision5 Software	1

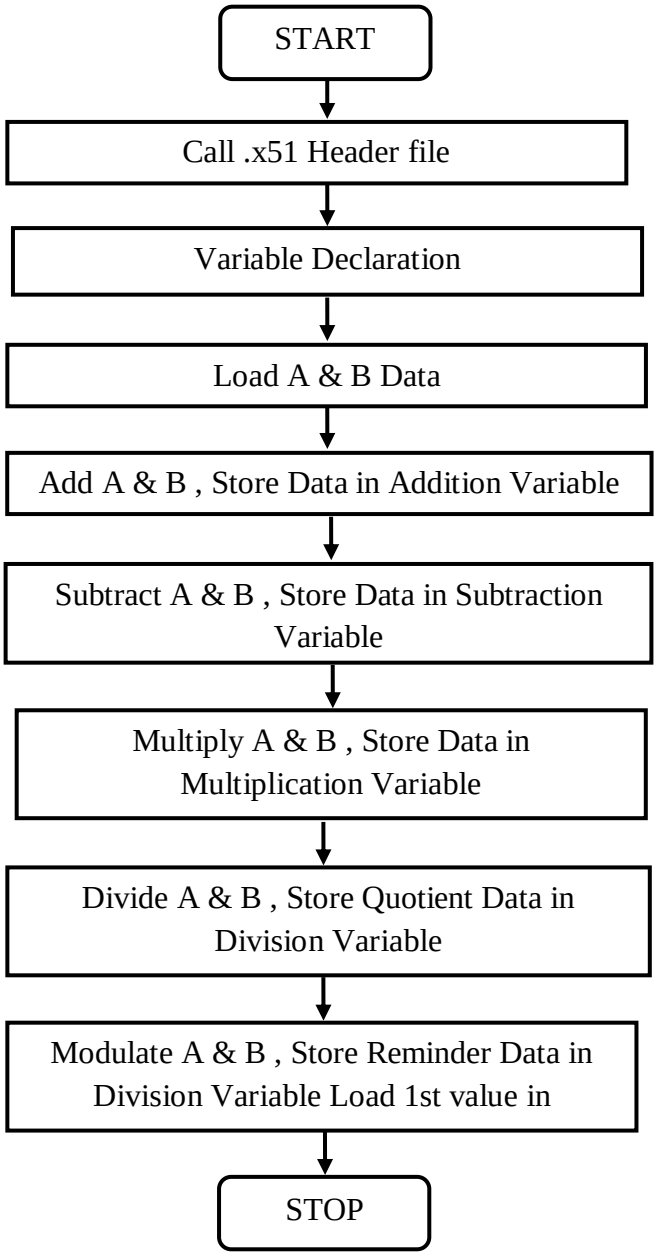
ALGORITHM:

1. Start the program.
2. Call x51 header file
3. Declare Variables
3. Load 1st value in A
4. Load 2nd value in B
5. Add A and B values and result store in the Addition variable
6. Subtract A and B values and result store in the Subtraction variable
7. Multiply A and B values and result store in the Multiplication variable
8. Divide A and B values and resultant Quotient store in the Division variable
9. Modulation A and B values and resultant Reminder store in the Modulation variable
10. Stop the program.

PROGRAM

PROGRAM	COMMENT
#include<reg51.h>	X51.h Header file calling
void main() {	Main function with open bracket
unsigned int addition, multiplication; unsigned char a, b, subtract, division, modulation;	Variable Declaration
a=0x33; b=0x22;	Load 1st value in A Load 2nd value in B
addition=a+b;	Add A & B , Store Data in Addition Variable
subtract=a-b;	Subtract A & B , Store Data in Subtraction Variable
multiplication=a*b;	Multiply A & B , Store Data in Multiplication Variable
division=a/b;	Divide A & B , Store Quotient Data in Division Variable
modulation=a%b;	Modulate A & B , Store Reminder Data in Division Variable
while(1);	Stop Program
}	Main function with close bracket

FLOW CHART


START
Call .x51 Header file
Variable Declaration
Load A & B Data
Add A & B , Store Data in Addition Variable
Subtract A & B , Store Data in Subtraction Variable
Multiply A & B , Store Data in Multiplication Variable
Divide A & B , Store Quotient Data in Division Variable
Modulate A & B , Store Reminder Data in Division Variable Load 1st value in
STOP

PROCEDURE:

1. Create **New Folder** & Rename as **ProjectFolder.** EX: Arithmetic
2. Save **ProjectFolder** where we desire.
3. Open **Keil uVision5** Software.
4. Click **Project >> New uVision Project**

   Create **Project** in **ProjectFolder** and Click **Save.** EX : arithmeticfuncion
5. Click **File >> New**

   Souce File Opended
6. Save Source File with **.c** Extension in **ProjectFolder.** EX: main.c
7. In Project Section which is presented in Left of the Software Application,

   Right Click 'Source Group 1' ....

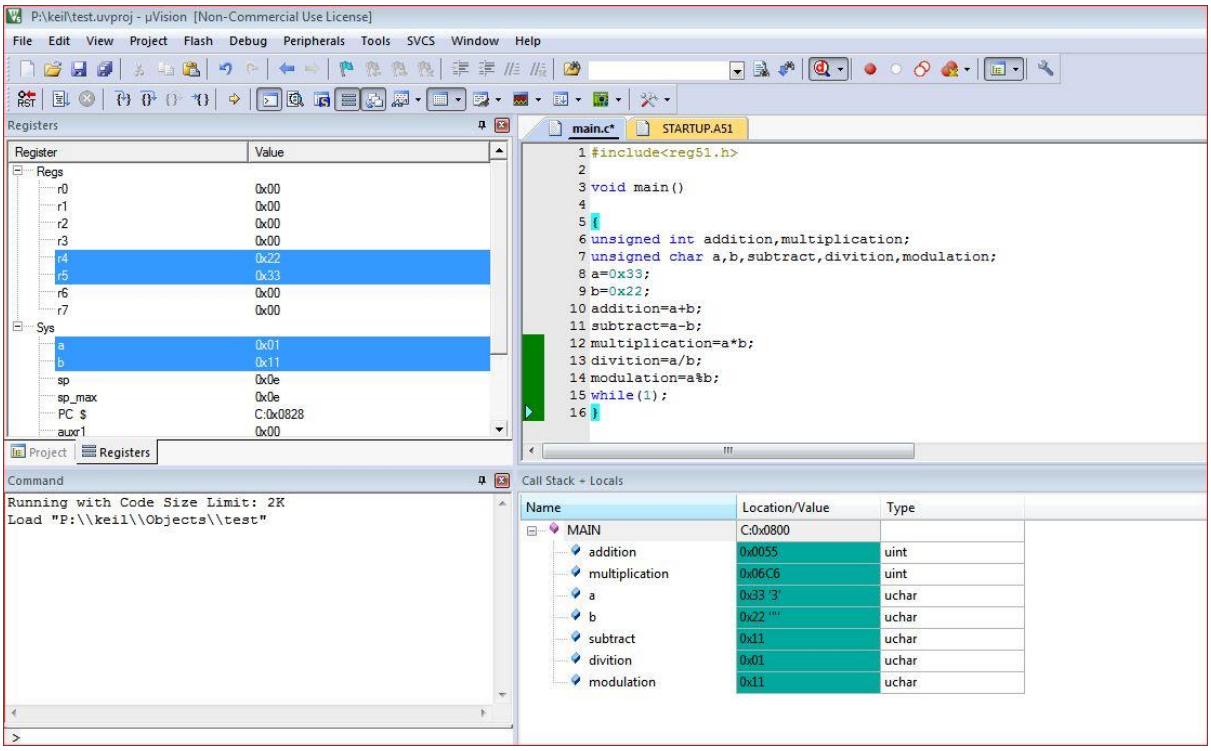
   Select the **Source File** in **ProjectFolder** , Click **Add** and Click **Close**.
8. Verify Source File is added or not by Expand the **'Source Group 1'** folder in Project Section.
9. Type the Program and Save it.
10. Click **Project >> Rebuilt all target files.** (If error presents debug the program & Click again **Project >> Rebuilt all target files** . Repeat this process until No Error and No Warning are presented)
11. Click **Debug >> Start / Stop Debug Session.**

Register Section, Command Section and Call Stacks + locals Sections are opened.

12. Click **Debug >> Run**
13. Click **Debug >> Stop**
14. Result are shown in **Call Stacks + locals** Section.
15. Verify the Result & Stop the program.

OUTPUT:	
Addition : **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
Addition= 55h = 0101 0101$_2$	
Subtraction : **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
Subtraction= 11h = 0001 0001$_2$	
Multiplication : **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
RESULT = 06 C6h = 0000 0110 1100 0110$_2$	
Division : **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
Quotient = Division = 01h = 0000 0001$_2$	
Modulation: **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
Reminder = Modulation = 11h = 0001 0001$_2$	

RESULT SCREEN SHOT



Result

Logical operation of two 8-Bit numbers in Embedded C Language

AIM:

To write and execute Embedded Language Program for Addition, Subtraction, Multiplication, Modulation and Division of two 8-Bit numbers using Keil uVision5.

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	Keil uVision5 Software	1

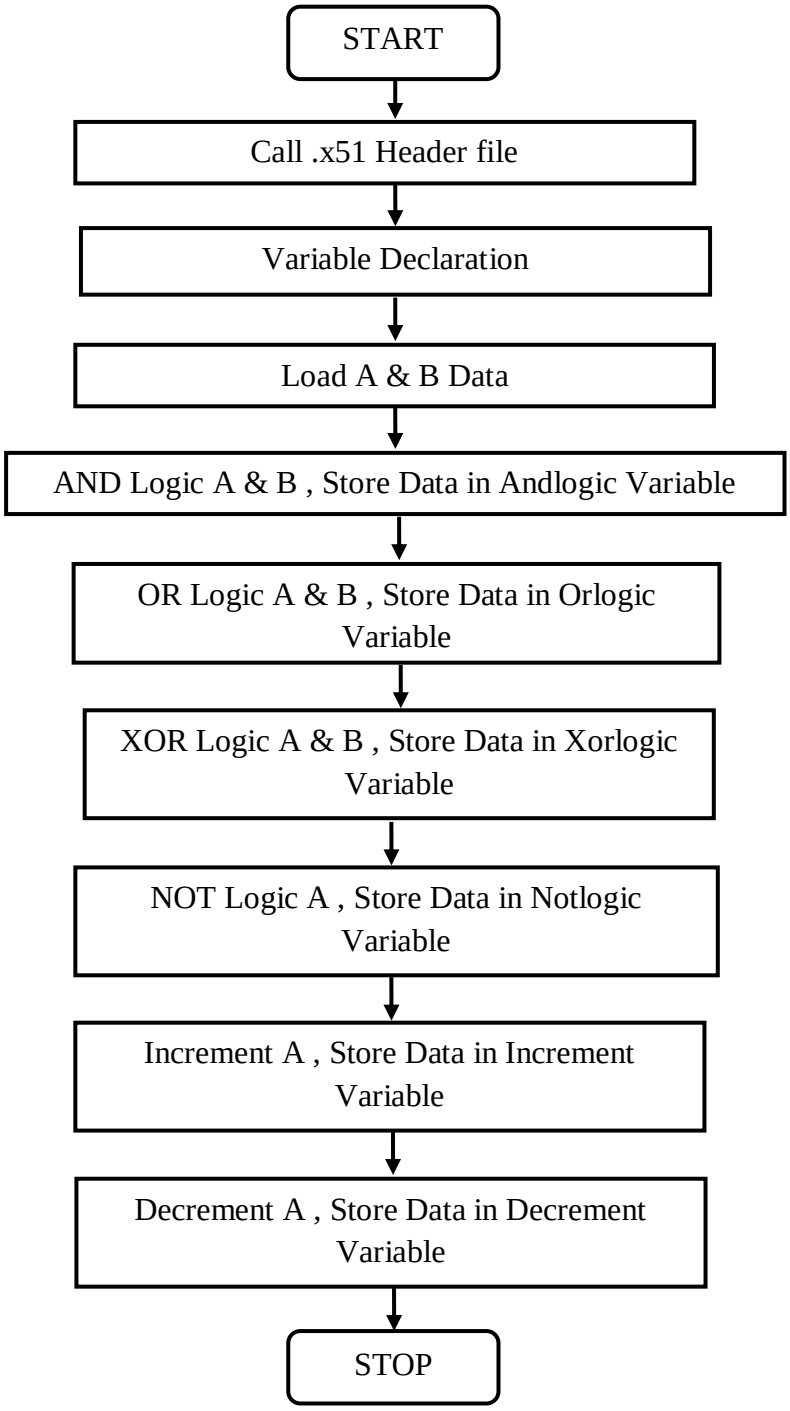
ALGORITHM:

1. Start the program.
2. Call x51 header file
3. Declare Variables
3. Load 1st value in A
4. Load 2nd value in B
5. Perform AND Logic A and B values and result store in the variable
6. Perform OR Logic A and B values and result store in the variable
7. Perform XOR Logic A and B values and result store in the variable
8. Perform NOT Logic A value and result store in the variable
9. Perform Increment Logic A value and result store in the variable
10. Perform Decrement Logic A value and result store in the variable
11. Stop the program.

PROGRAM

PROGRAM	COMMENT
#include<reg51.h>	X51.h Header file calling
void main() {	Main function with open bracket
unsigned int addition, multiplication; unsigned char a, b, subtract, division, modulation;	Variable Declaration
a=0x33; b=0x22;	Load 1st value in A Load 2nd value in B
andlogic=a&b;	AND Logic A & B , Store Data in Andlogic Variable
orlogic=a | b;	OR Logic A & B , Store Data in Orlogic Variable
xorlogic=a^b;	XOR Logic A & B , Store Data in Xorlogic Variable
notlogic=~a;	NOT Logic A , Store Data in Notlogic Variable
increment= a+0x01;	Increment A , Store Data in Increment Variable
decrement =a-0x01;	Decrement A , Store Data in Decrement Variable
while(1);	Stop Program
}	Main function with close bracket

FLOW CHART


START
Call .x51 Header file
Variable Declaration
Load A & B Data
AND Logic A & B , Store Data in Andlogic Variable
OR Logic A & B , Store Data in Orlogic Variable
XOR Logic A & B , Store Data in Xorlogic Variable
NOT Logic A , Store Data in Notlogic Variable
Increment A , Store Data in Increment Variable
Decrement A , Store Data in Decrement Variable
STOP

PROCEDURE:

1. Create **New Folder** & Rename as **ProjectFolder.** EX: Arithmetic
2. Save **ProjectFolder** where we desire.
3. Open **Keil uVision5** Software.
4. Click **Project >> New uVision Project**

   Create **Project** in **ProjectFolder** and Click **Save.** EX : arithmeticfuncion
5. Click **File >> New**

   Souce File Opended
6. Save Source File with **.c** Extension in **ProjectFolder.** EX: main.c
7. In Project Section which is presented in Left of the Software Application,

   Right Click 'Source Group 1' ....

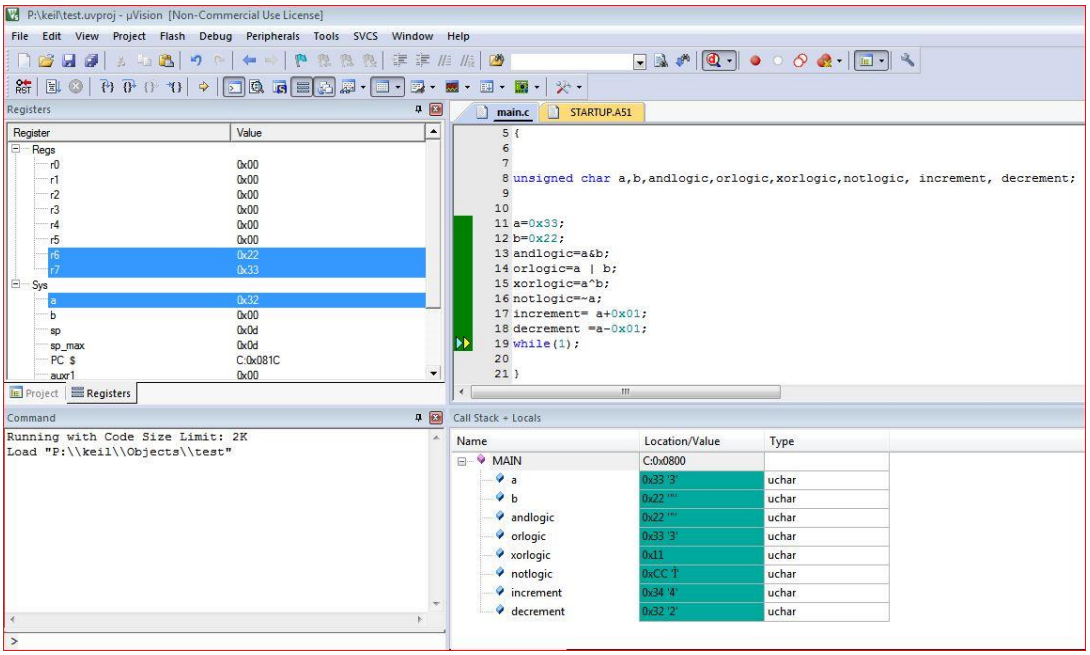
   Select the **Source File** in **ProjectFolder** , Click **Add** and Click **Close**.
8. Verify Source File is added or not by Expand the **'Source Group 1'** folder in Project Section.
9. Type the Program and Save it.
10. Click **Project >> Rebuilt all target files.** (If error presents debug the program & Click again **Project >> Rebuilt all target files** . Repeat this process until No Error and No Warning are presented)
11. Click **Debug >> Start / Stop Debug Session.**

Register Section, Command Section and Call Stacks + locals Sections are opened.

12. Click **Debug >> Run**
13. Click **Debug >> Stop**
14. Result are shown in **Call Stacks + locals** Section.
15. Verify the Result & Stop the program.

OUTPUT:	
AND: **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
ANDLOGIC= 22h = 0010 0010$_2$	
OR: **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
ORLOGIC= 33h = 0011 0011$_2$	
XOR : **A = 33h = 0011 0011$_2$** **B = 22h = 0010 0010$_2$**	
XORLOGIC = 11h = 0001 0001$_2$	
NOT: **A = 33h = 0011 0011$_2$**	
NOTLOGIC = CCh = 1100 1100$_2$	
INCREMENT: **A = 33h = 0011 0011$_2$**	
INC = 34h = 0011 0100$_2$	
DECREMENT: **A = 33h = 0011 0011$_2$**	
DEC = 32h = 0011 0010$_2$	

RESULT SCREEN SHOT



Result

Interface and Program LED Array with Arduino Using Arduino platform

AIM:

To interface the 1X4 LED array with Arduino using Arduino IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Arduino Nano unit • LED Array unit	1
3	Jumper Wires	As Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Define GPIO Pins
3. Define GPIO Pin Mode
4. if Infinite loop True:

    ON LED_1

    Delay 1 Second

    ON LED_2

    Delay 1 Second

    ON LED_3

    Delay 1 Second

    ON LED_4

    Delay 1 Second

    OFF LED_1

OFF LED_2

OFF LED_3

OFF LED_4

Delay 1 Second

5. else

Stop Program

PROGRAM

PROGRAM	COMMENT
int led1 = 2;	LED_1 connected with D2
int led2 = 3;	LED_2 connected with D3
int led3 = 4;	LED_3 connected with D4
int led4 = 5;	LED_4 connected with D5
void setup() {	Set up Loop Start
pinMode(led1 , OUTPUT);	Configure LED_1 as Output
pinMode(led2 , OUTPUT);	Configure LED_2 as Output
pinMode(led3 , OUTPUT);	Configure LED_3 as Output
pinMode(led4 , OUTPUT);	Configure LED_4 as Output
}	Set up Loop Close
void loop() {	Infinite Loop Start
digitalWrite(led1, HIGH);	LED_1 ON
delay(1000);	Delay 1 Second
digitalWrite(led2, HIGH);	LED_2 ON
delay(1000);	Delay Second
digitalWrite(led3, HIGH);	LED_3 ON
delay(1000);	Delay 1 Second
digitalWrite(led4, HIGH);	LED_4 ON
delay(1000);	Delay 1 Second
digitalWrite(led1, LOW);	LED_1 OFF

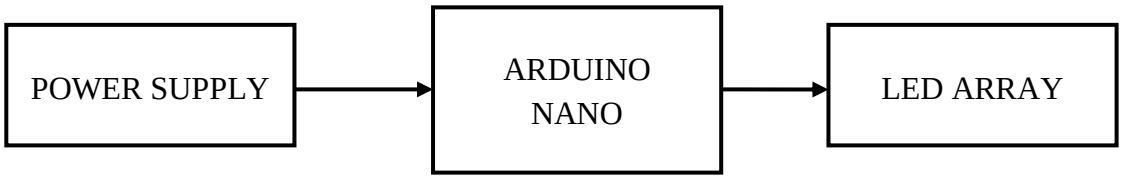
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
	LED_2 OFF
	LED_3 OFF
	LED_4 OFF

delay(1000);	Delay 1 Second
}	Infinite Loop Close

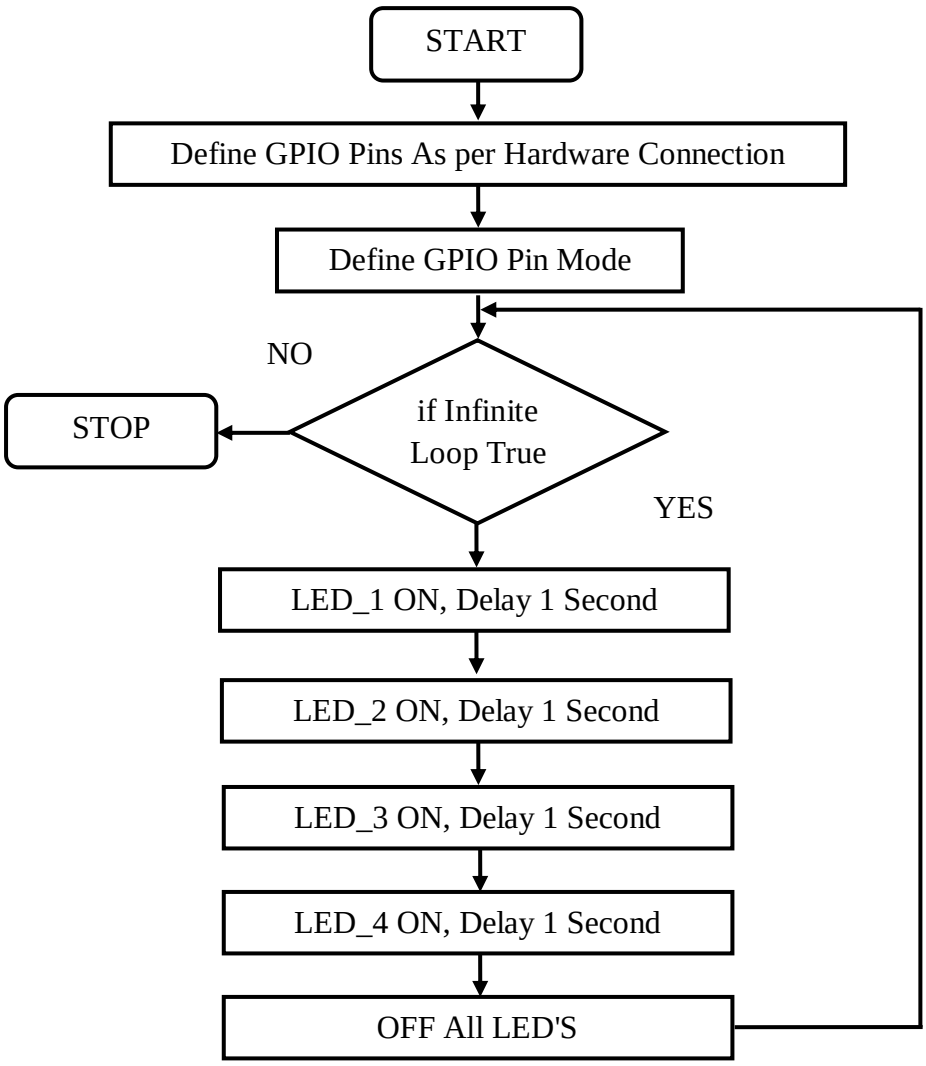
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	ARDUINO	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LED_1 (L1)	D2	OUTPUT
4	LED_2 (L2)	D3	OUTPUT
5	LED_3 (L3)	D4	OUTPUT
6	LED_4 (L4)	D5	OUTPUT

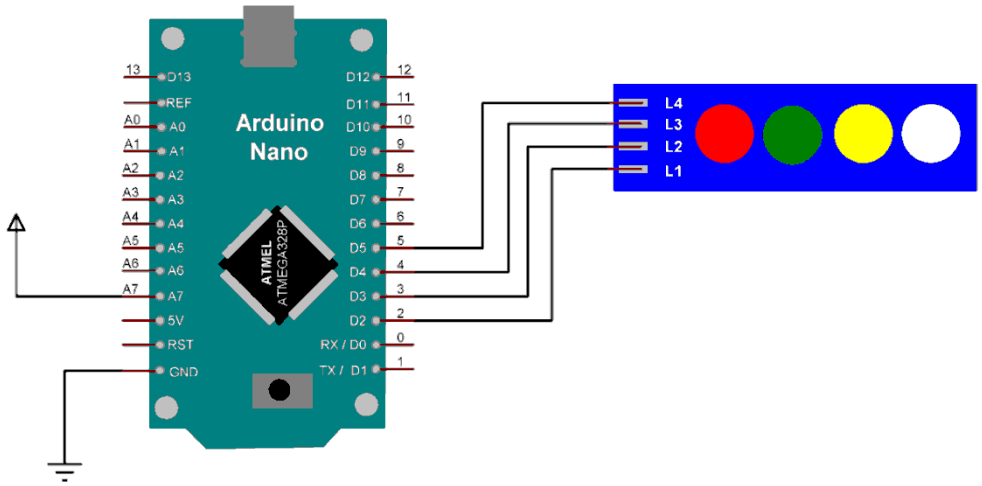
BLOCK DIAGRAM



FLOW CHART



CIRCUIT DIAGRAM



PROCEDURE:

1. Open **ARDUINO IDE** Software
2. Click **File >> New** or **CTRL+N** to create New Project.
3. To Select Board Click **Tools >> Board > Arduino AVR Boards >> Arduino Nano**
4. To select Processor Click **Tools >> Processor >> ATmega328P (Old Bootloader)**
5. To select Programmer Click **Tools >> Programmer >> AVRISP mkII**
6. Click **File >> Save** or **CTRL+S** to Save Project where we Desire.
7. Type the Program.
8. Click **Sketch >> Verify / Compile** or **CTRL+R** to compile the program (If any error present debug the program)
9. Connect the Arduino Nano with PC / Laptop by USB Cable
10. Select the COM PORT by Click **Tools >> Port** and Select Arduino's COM PORT Number.

Note : If COM PORT Number is Don't Know:

- Right Click My Computer Icon and Select Properties
- Click Device Manager
- Expand Ports (COM & LPT). COM PORT Number will be shown

11. Click **Sketch >> Upload** or **CTRL+U** to upload the program to Arduino Nano.
12. Switch On the corresponding Units we need and Verify the Program.

Result

Interface and program I2C LCD with Arduino Using Arduino Platform

AIM:

To interface the I2C LCD with Arduino using Arduino IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Arduino Nano Unit • I2C LCD Unit	1
3	Jumper Wires	AS Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Call I2C LCD Library
3. Initiate LCD
4. if Infinite loop True:

   Set LCD Cursor (0,0) Position 0 and Column 1

   Print **HELLO WORLD**

   Set LCD Cursor (0,1) Position 0 and Column 2

   Print *****

   Delay 2 Second

   Clear LCD

   Delay 2 Second
5. else

   Stop Program

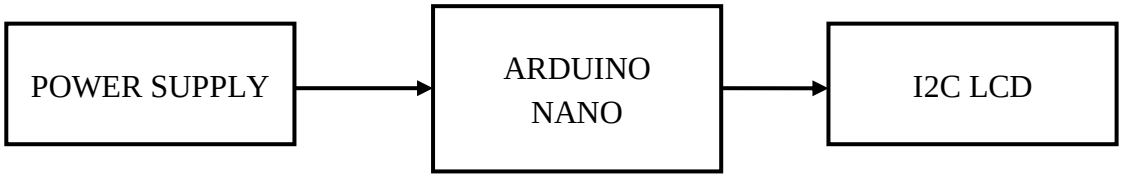
PROGRAM

PROGRAM	COMMENT
#include <Wire.h>	Call 1 wire header file
#include <LiquidCrystal_I2C.h>	Call I2C LCD header file
LiquidCrystal_I2C lcd(0x27,16,2);	Define I2C LCD Address
void setup() {	Set up Loop Start
lcd.init();	Initialize the LCD
lcd.backlight();	LCD Back light ON
lcd.clear();	Clear LCD
}	Set up Loop Close
void loop() {	Infinite Loop Start
lcd.setCursor(0,0);	Set LCD Cursor (0,0) Position 0 and Column 1
lcd.print(" HELLO WORLD");	Print **HELLO WORLD**
lcd.setCursor(0,1);	Set LCD Cursor (0,1) Position 0 and Column 2
lcd.print(" *****");	Print *****
delay(2000);	Delay 2 Second
lcd.clear();	Clear LCD
delay(2000);	Delay 2 Second
}	Infinite Loop Close

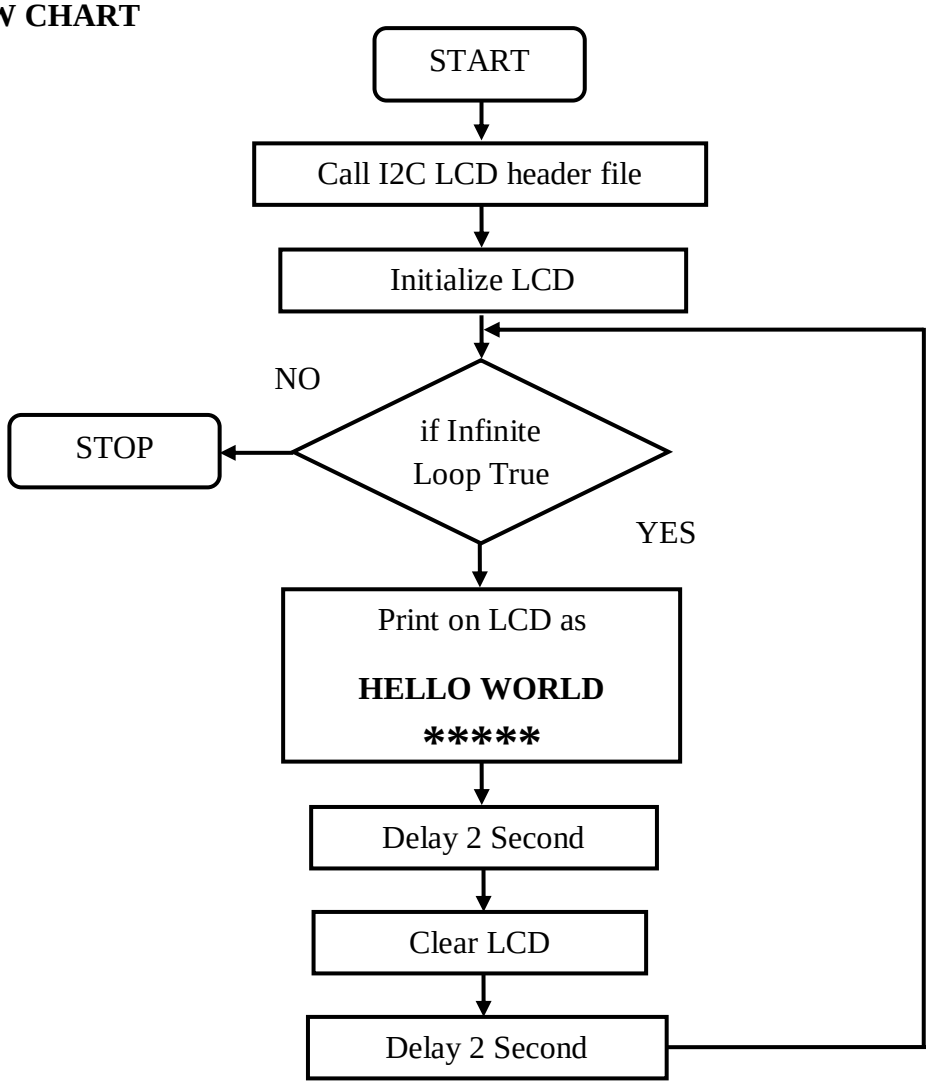
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	ARDUINO	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LCD (SDA)	A4	Serial Data
4	LCD (SCL)	A5	Serial Clock

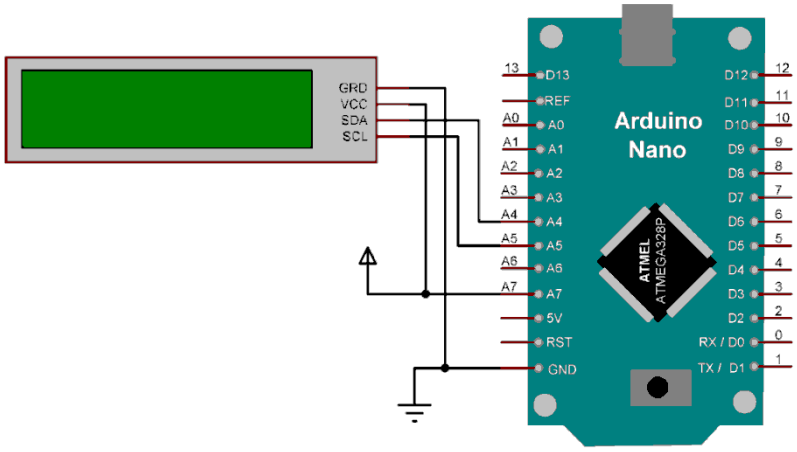
BLOCK DIAGRAM



FLOW CHART



CIRCUIT DIAGRAM



PROCEDURE:

1. Open **ARDUINO IDE** Software
2. Click **File >> New** or **CTRL+N** to create New Project.
3. To Select Board Click **Tools >> Board > Arduino AVR Boards >> Arduino Nano**
4. To select Processor Click **Tools >> Processor >> ATmega328P (Old Bootloader)**
5. To select Programmer Click **Tools >> Programmer >> AVRISP mkII**
6. Click **File >> Save** or **CTRL+S** to Save Project where we Desire.
7. Type the Program.
8. Click **Sketch >> Verify / Compile** or **CTRL+R** to compile the program (If any error present debug the program)
9. Connect the Arduino Nano with PC / Laptop by USB Cable
10. Select the COM PORT by Click **Tools >> Port** and Select Arduino's COM PORT Number.

Note : If COM PORT Number is Don't Know:

- Right Click My Computer Icon and Select Properties
- Click Device Manager
- Expand Ports (COM & LPT). COM PORT Number will be shown

11. Click **Sketch >> Upload** or **CTRL+U** to upload the program to Arduino Nano.
12. Switch On the corresponding Units we need and Verify the Program.

Result

Interface and program Bluetooth Communication with Arduino using Arduino Platform

AIM:

To interface the Bluetooth with Arduino using Arduino IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Arduino Nano Unit • HC-05 Bluetooth Unit	1
3	Jumper Wires	As Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Variable declaration
3. Initialize Bluetooth
4. Send **Welcome Message**
5. if Infinite loop True:

   if Bluetooth Data Available

   Receive Data And Send Again to Bluetooth Module
6. else

   Stop Program

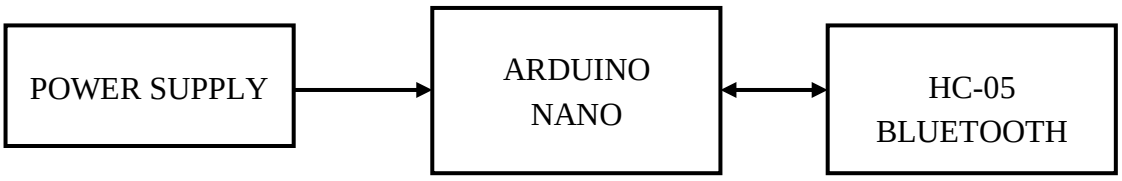
PROGRAM

PROGRAM	COMMENT
String read_data;	Variable Declaration
void setup() {	Set up Loop Start
Serial.begin(9600);	Set Baud Rate as 9600
}	Set up Loop Close
void loop() {	Infinite Loop Start
while (Serial.available()) {	Check if there is an available byte to read
delay(10);	Delay added to make thing stable
char c = Serial.read();	Conduct a serial read
read_data+= c;	Build the string
}	
if (read_data.length() > 0)	Check Full Word is Received
{	
Serial.println(read_data);	Send Data to Bluetooth
}	
read_data="";	Reset the variable
}	Infinite Loop Close

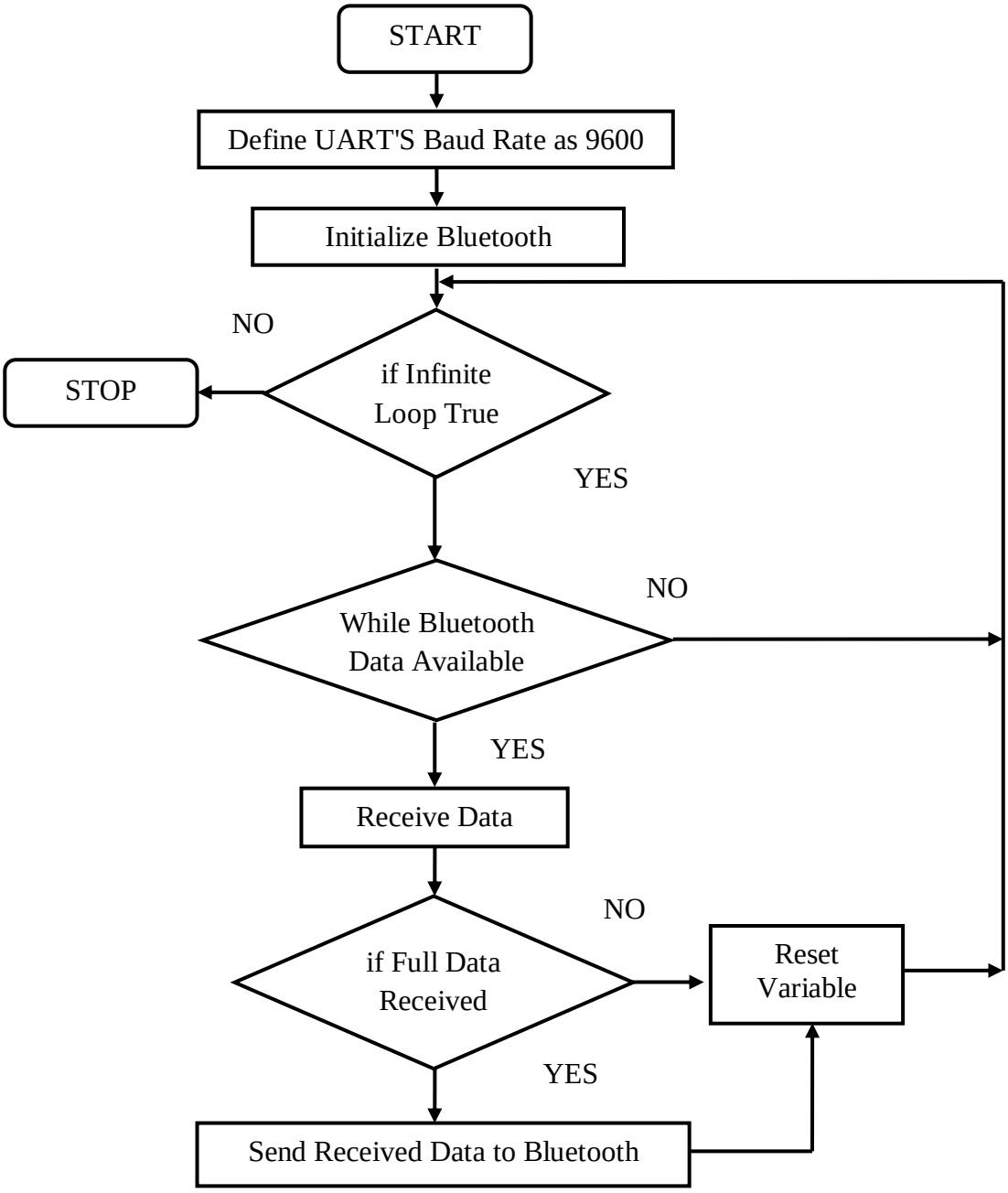
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	ARDUINO	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	HC-05_TX	RX	DATA RECIEVE
4	HC-05_RX	TX	DATA TRANSMIT

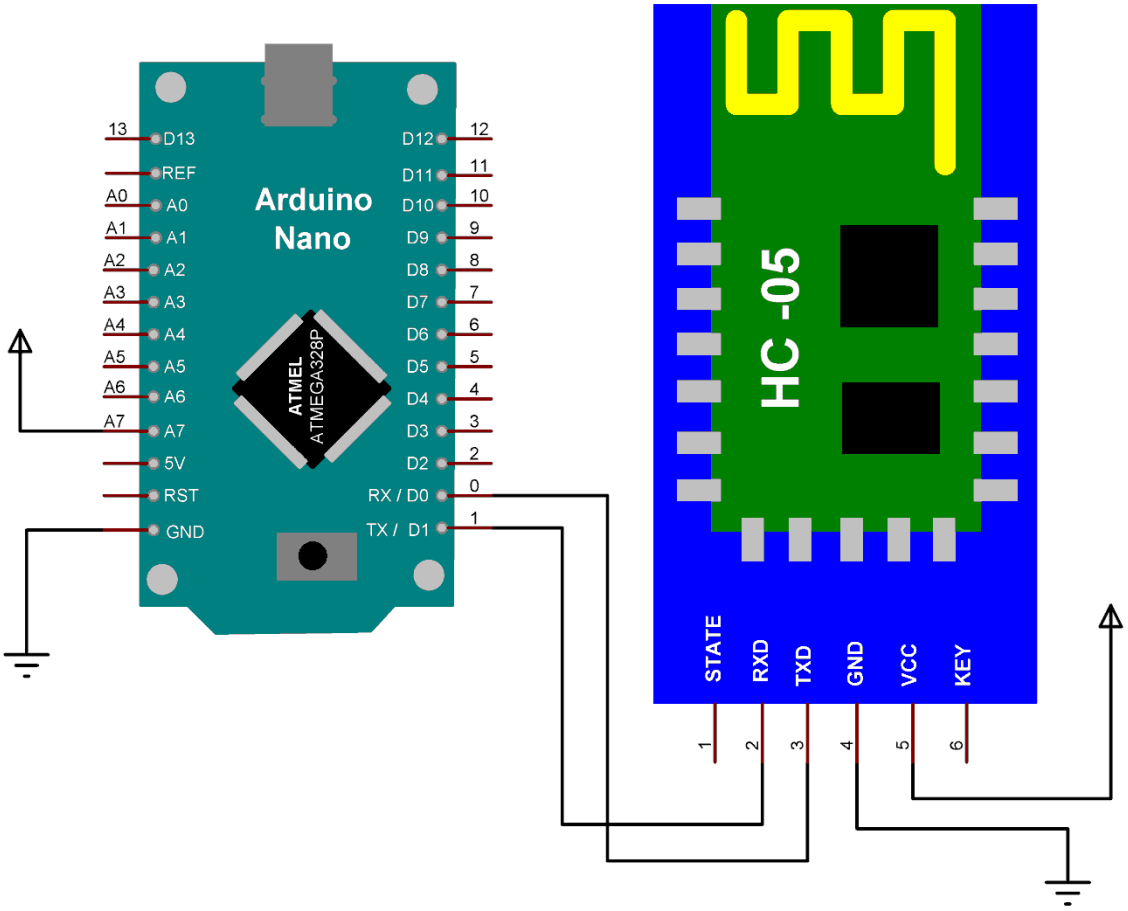
BLOCK DIAGRAM



FLOW CHART



CIRCUIT DIAGRAM



PROCEDURE:

1. Open **ARDUINO IDE** Software
2. Click **File >> New** or **CTRL+N** to create New Project.
3. To Select Board Click **Tools >> Board > Arduino AVR Boards >> Arduino Nano**
4. To select Processor Click **Tools >> Processor >> ATmega328P (Old Bootloader)**
5. To select Programmer Click **Tools >> Programmer >> AVRISP mkII**
6. Click **File >> Save** or **CTRL+S** to Save Project where we Desire.
7. Type the Program.
8. Click **Sketch >> Verify / Compile** or **CTRL+R** to compile the program (If any error present debug the program)

9. Connect the Arduino Nano with PC / Laptop by USB Cable

10. Select the COM PORT by Click **Tools >> Port** and Select Arduino's COM PORT Number.

Note : If COM PORT Number is Don't Know:

- Right Click My Computer Icon and Select Properties
- Click Device Manager
- Expand Ports (COM & LPT). COM PORT Number will be shown

11. Click **Sketch >> Upload** or **CTRL+U** to upload the program to Arduino Nano.

12. Switch On the corresponding Units we need and Verify the Program.

Result

Interface and program Zigbee Communication with Arduino using Arduino Platform

AIM:

To interface the Zigbee with Arduino using Arduino IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Arduino Nano Unit • Zigbee Unit	1
3	Jumper Wires	As Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Variable declaration
3. Initialize Zigbee
4. Send **Welcome Message**
5. if Infinite loop True:

   if Zigbee Data Available

   Receive Data And Send Again to Zigbee Module
6. else

   Stop Program

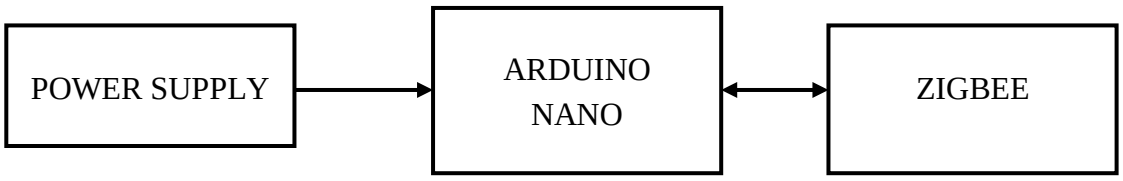
PROGRAM

PROGRAM	COMMENT
String read_data;	Variable Declaration
void setup() {	Set up Loop Start
Serial.begin(9600);	Set Baud Rate as 9600
}	Set up Loop Close
void loop() {	Infinite Loop Start
while (Serial.available()) {	Check if there is an available byte to read
delay(10);	Delay added to make thing stable
char c = Serial.read();	Conduct a serial read
read_data+= c;	Build the string
}	
if (read_data.length() > 0)	Check Full Word is Received
{	
Serial.println(read_data);	Send Data to Zigbee
}	
read_data="";	Reset the variable
}	Infinite Loop Close

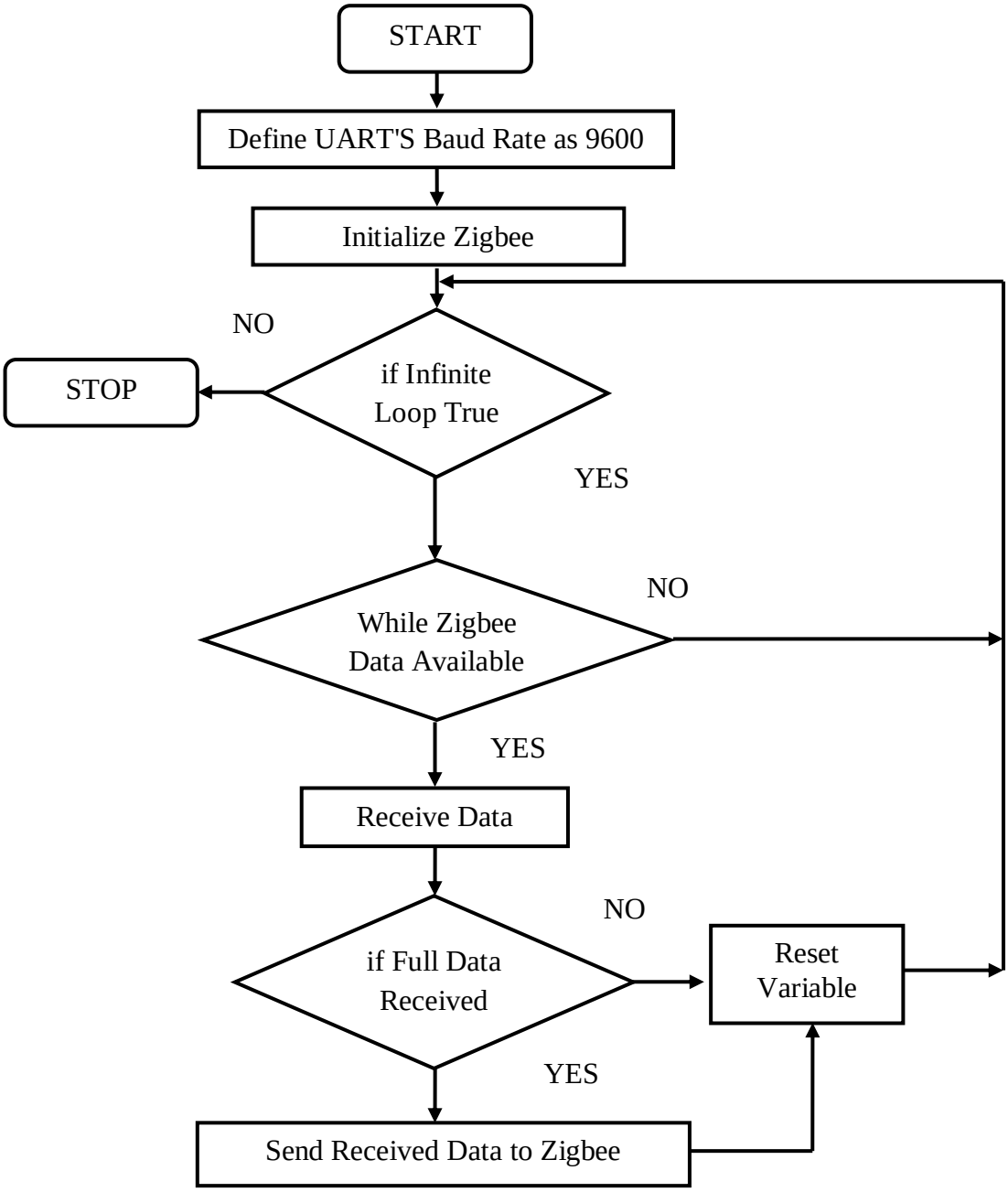
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	ARDUINO	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	Zigbee_TX	RX	DATA RECIEVE
4	Zigbee_RX	TX	DATA TRANSMIT

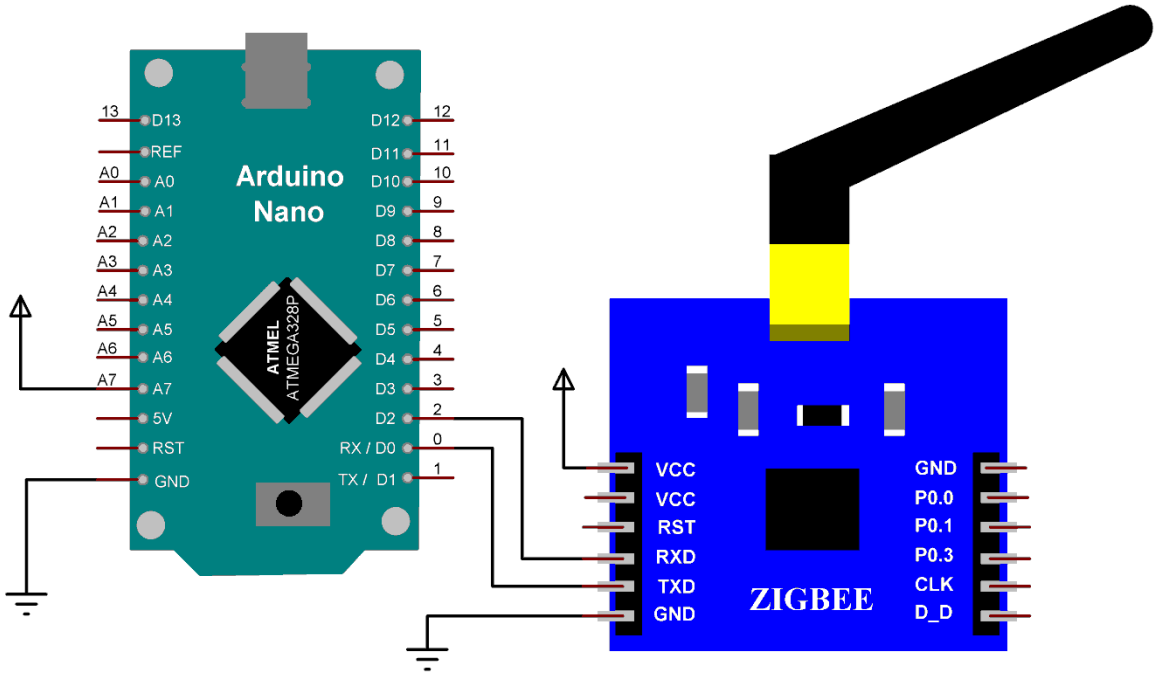
BLOCK DIAGRAM



FLOW CHART



CIRCUIT DIAGRAM



PROCEDURE:

1. Open **ARDUINO IDE** Software
2. Click **File >> New** or **CTRL+N** to create New Project.
3. To Select Board Click **Tools >> Board > Arduino AVR Boards >> Arduino Nano**
4. To select Processor Click **Tools >> Processor >> ATmega328P (Old Bootloader)**
5. To select Programmer Click **Tools >> Programmer >> AVRISP mkII**
6. Click **File >> Save** or **CTRL+S** to Save Project where we Desire.
7. Type the Program.
8. Click **Sketch >> Verify / Compile** or **CTRL+R** to compile the program (If any error present debug the program)
9. Connect the Arduino Nano with PC / Laptop by USB Cable
10. Select the COM PORT by Click **Tools >> Port** and Select Arduino's COM PORT Number.

Note : If COM PORT Number is Don't Know:

- Right Click My Computer Icon and Select Properties
- Click Device Manager
- Expand Ports (COM & LPT). COM PORT Number will be shown

11. Click **Sketch >> Upload** or **CTRL+U** to upload the program to Arduino Nano.
12. Switch On the corresponding Units we need and Verify the Program.

Result

Interface and program GSM Communication with Arduino using Arduino Platform

AIM:

To interface the GSM with Arduino and Send SMS using Arduino IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Arduino Nano Unit • GSM Unit	1
3	Jumper Wires	As Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Variable declaration
3. Wait 20 Seconds
4. Initialize GSM
5. Check SIM Signal Strength
6. Configure GSM as Text Mode
7. Define SMS Receive Mobile Number
8. Define SMS Message Content
9. Send **SMS**
10. Stop Program

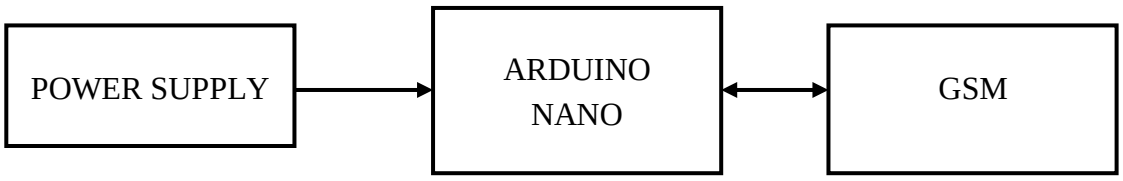
PROGRAM

PROGRAM	COMMENT
void setup() {	Set up Loop Start
Serial.begin(9600);	Set Baud Rate as 9600
delay(20000);	Delay 1 Second
Serial.println("AT");	Handshake test is successful
delay(3000);	
Serial.println("AT+CMGF=1");	Configuring TEXT mode
delay(3000);	
Serial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");	ZZ is Country Code xxxxxxxxxx is Mobile Number ex: +918903732238
delay(3000);	
Serial.print("hello world");	Text content
delay(3000);	
Serial.write(26);	Send SMS Command
delay(3000);	
}	Set up Loop Close
void loop() {	Infinite Loop Start
delay(1000);	Delay 1 Second
}	Infinite Loop Close

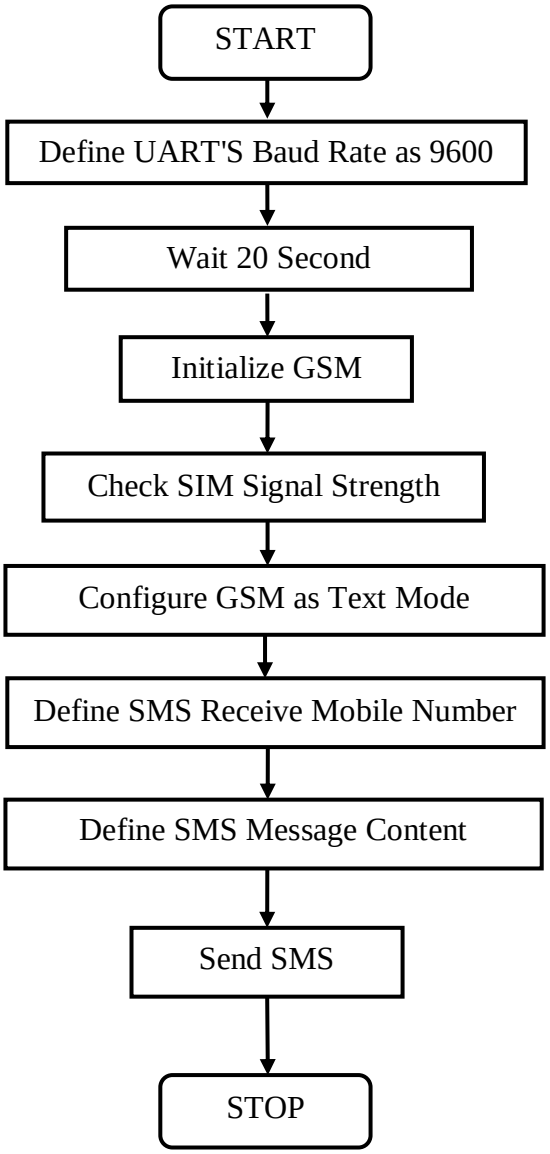
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	ARDUINO	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	GSM_TX	RX	DATA RECIEVE
4	GSM_RX	TX	DATA TRANSMIT

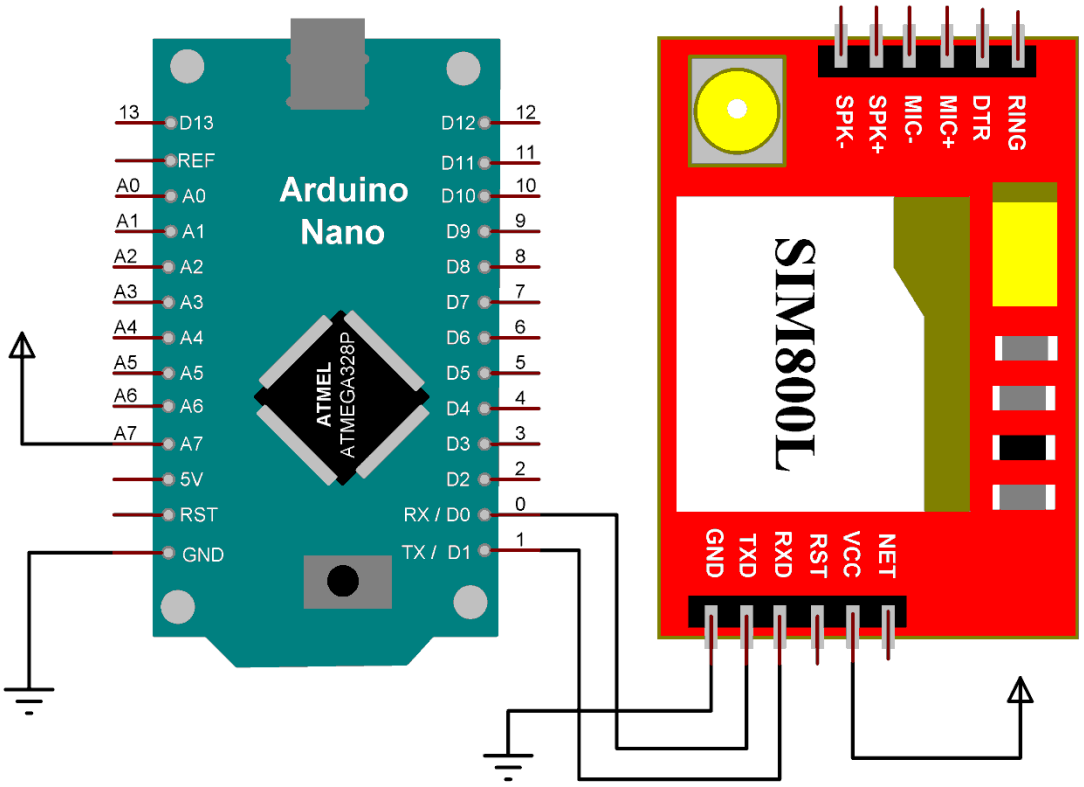
BLOCK DIAGRAM



FLOW CHART



CIRCUIT DIAGRAM



PROCEDURE:

1. Open **ARDUINO IDE** Software
2. Click **File >> New** or **CTRL+N** to create New Project.
3. To Select Board Click **Tools >> Board > Arduino AVR Boards >> Arduino Nano**
4. To select Processor Click **Tools >> Processor >> ATmega328P (Old Bootloader)**
5. To select Programmer Click **Tools >> Programmer >> AVRISP mkII**
6. Click **File >> Save** or **CTRL+S** to Save Project where we Desire.
7. Type the Program.
8. Click **Sketch >> Verify / Compile** or **CTRL+R** to compile the program (If any error present debug the program)
9. Connect the Arduino Nano with PC / Laptop by USB Cable

10. Select the COM PORT by Click **Tools >> Port** and Select Arduino's COM PORT Number.

Note : If COM PORT Number is Don't Know:

- Right Click My Computer Icon and Select Properties
- Click Device Manager
- Expand Ports (COM & LPT). COM PORT Number will be shown

11. Click **Sketch >> Upload** or **CTRL+U** to upload the program to Arduino Nano.

12. Switch On the corresponding Units we need and Verify the Program.

Result

Interface LED Array With Raspberry Pi Platform And Python Programming

AIM:

To interface the 1X4 LED array with Raspberry Pi using Thoony Python IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Raspberry Pi unit • LED Array unit	1
3	Jumper Wires	As Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Define GPIO Pins
3. Define GPIO Pin Mode
4. if Infinite loop True:

    ON LED_1

    Delay 1 Second

    ON LED_2

    Delay 1 Second

    ON LED_3

    Delay 1 Second

    ON LED_4

    Delay 1 Second

    OFF LED_1

OFF LED_2

OFF LED_3

OFF LED_4

Delay 1 Second

5. else

Stop Program

PROGRAM

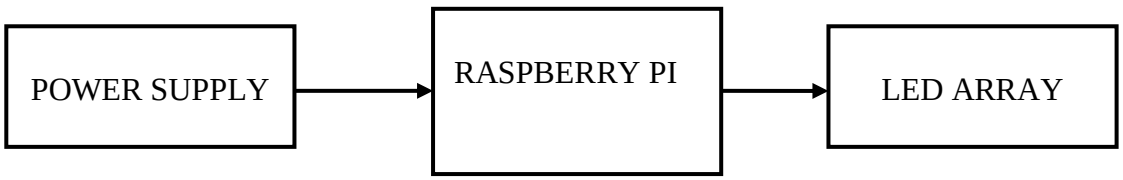
PROGRAM	COMMENT
import RPi.GPIO as GPIO	import GPIO library
import time	import Time Library
GPIO.setwarnings(False)	Warning Off for GPIO pins
LED_PIN1 = 18	LED_1 connected with GPIO 18
LED_PIN2 = 23	LED_2 connected with GPIO 23
LED_PIN3 = 24	LED_3 connected with GPIO 24
LED_PIN4 = 25	LED_4 connected with GPIO 25
GPIO.setmode(GPIO.BCM)	Define as Physical numbering
GPIO.setup(LED_PIN1, GPIO.OUT)	Configure LED_1 as Output
GPIO.setup(LED_PIN2, GPIO.OUT)	Configure LED_2 as Output
GPIO.setup(LED_PIN3, GPIO.OUT)	Configure LED_3 as Output
GPIO.setup(LED_PIN4, GPIO.OUT)	Configure LED_4 as Output
while True:	Infinite Loop Start
GPIO.output(LED_PIN1, GPIO.HIGH)	LED_1 ON
time.sleep(1)	Delay 1 Second
GPIO.output(LED_PIN2, GPIO.HIGH)	LED_2 ON
time.sleep(1)	Delay Second
GPIO.output(LED_PIN3, GPIO.HIGH)	LED_3 ON
time.sleep(1)	Delay 1 Second
GPIO.output(LED_PIN4, GPIO.HIGH)	LED_4 ON
time.sleep(1)	Delay 1 Second
GPIO.output(LED_PIN4, GPIO.LOW)	LED_1 OFF

GPIO.output(LED_PIN3, GPIO.LOW)	LED_2 OFF
GPIO.output(LED_PIN2, GPIO.LOW)	LED_3 OFF
GPIO.output(LED_PIN1, GPIO.LOW)	LED_4 OFF
except:	Infinite Loop Close
GPIO.cleanup()	Reset GPIO Pins

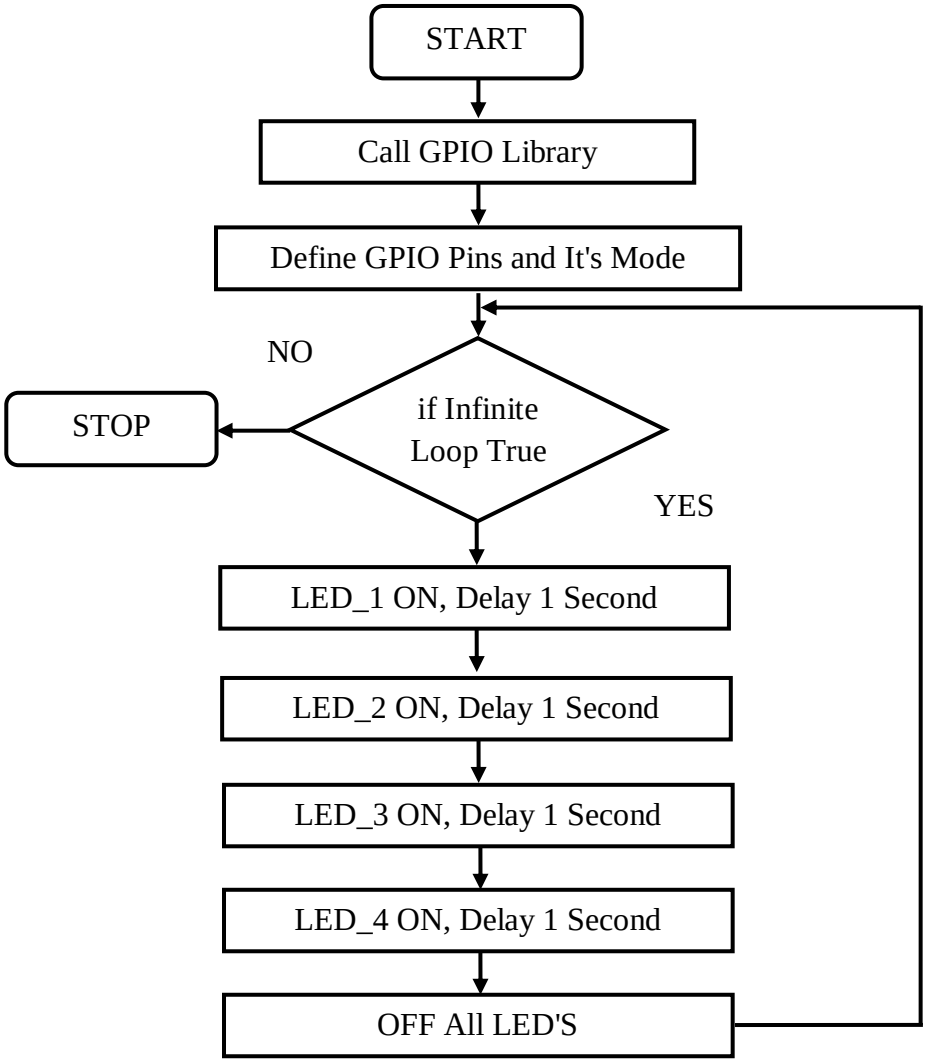
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	RASPBERRY PI	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LED_1 (L1)	GPIO 18	OUTPUT
4	LED_2 (L2)	GPIO 23	OUTPUT
5	LED_3 (L3)	GPIO 24	OUTPUT
6	LED_4 (L4)	GPIO 25	OUTPUT

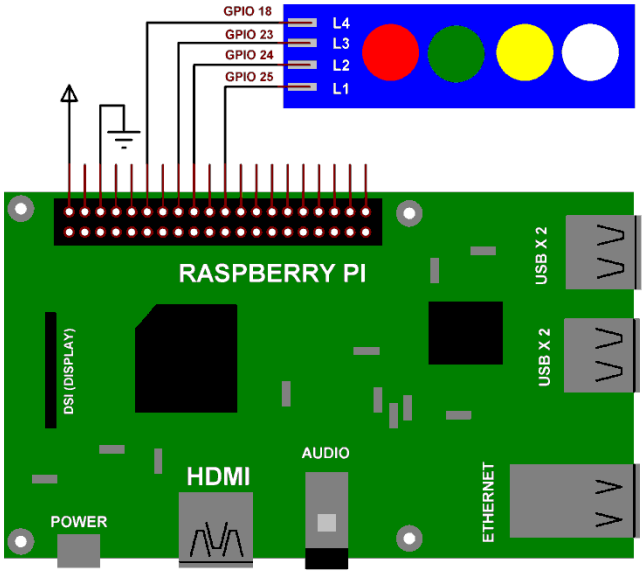
BLOCK DIAGRAM



FLOW CHART



CIRCUIT DIAGRAM



TERMINAL COMMAND LINE

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get install rpi.gpio

PROCEDURE:

1. Start **Thonny** by clicking on the **Raspberry Pi icon** followed by **Programming > Thonny Python IDE.**
2. Write your program in the top pane, click **File > Save** as... to save it,
3. Make hardware connections
4. Click **Run** > Run current script to execute the program. Output will appear in the bottom interpreter pane.

Result

Interface LCD With Raspberry Pi Platform And Python Programming

AIM:

To interface the I2C LCD with Raspberry Pi using Thoony Python IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Raspberry Pi Unit • I2C LCD Unit	1
3	Jumper Wires	AS Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Call I2C LCD Library
3. Initiate LCD
4. if Infinite loop True:

    Set LCD Cursor (0,0) Position 0 and Column 1

    Print **HELLO WORLD**

    Set LCD Cursor (0,1) Position 0 and Column 2

    Print *****

    Delay 2 Second

    Clear LCD

    Delay 2 Second

5. else

    Stop Program

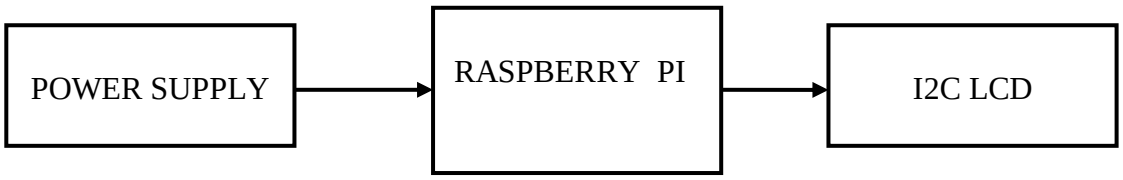
PROGRAM

PROGRAM	COMMENT
import lcddriver	Import LCD Library
from time import *	Import Delay Library
lcd = lcddriver.lcd()	Initialize LCD
try:	Test a block of code for errors
while True:	Infinite Loop Start
lcd.lcd_display_string(" HELLO WORLD", 1)	Print **HELLO WORLD** on LCD 1st Line
lcd.lcd_display_string(" ********", 2)	Print ********** on LCD 2nd Line
time.sleep(2)	Delay 2 Second
lcd.lcd_clear()	Clear LCD
time.sleep(2)	Delay 2 Second
except:	Block lets you handle the error
GPIO.cleanup()	Reset GPIO Pins

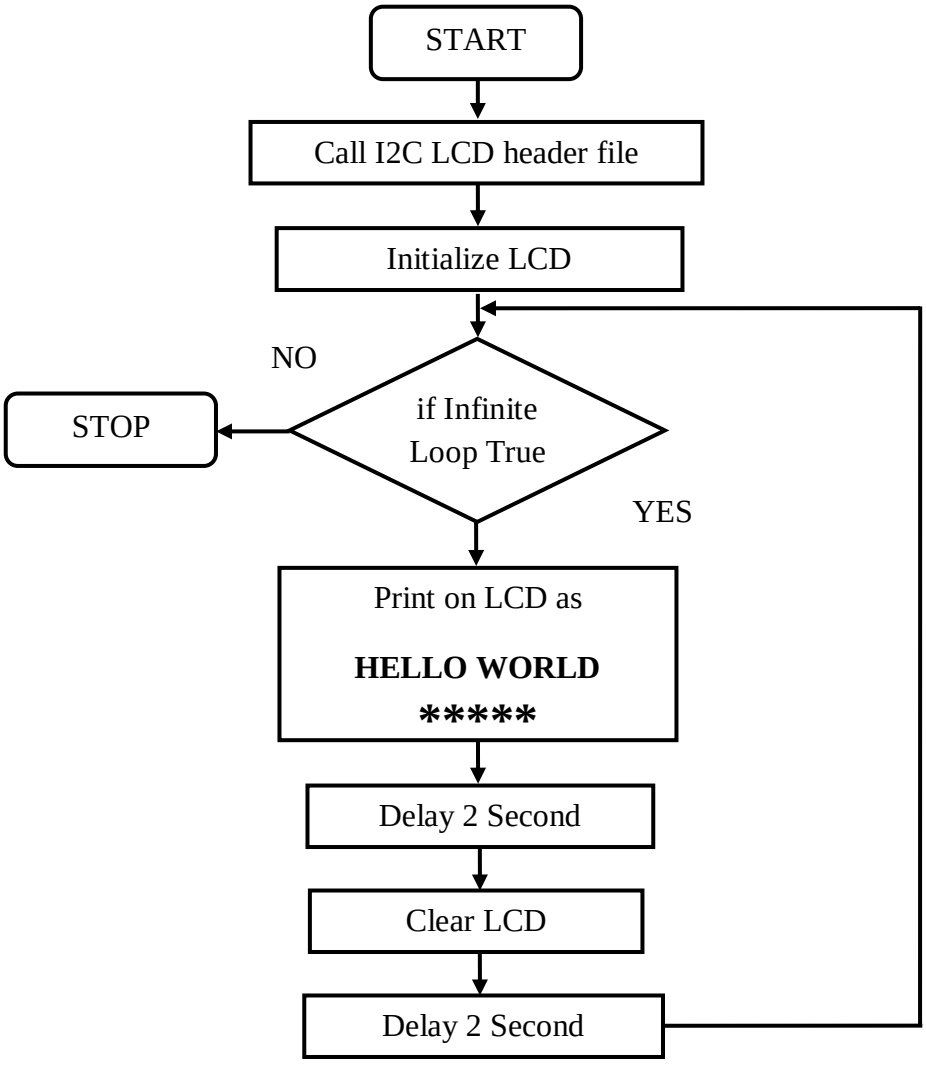
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	RASPBERRY PI	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LCD (SDA)	GPIO 2	Serial Data
4	LCD (SCL)	GPIO 3	Serial Clock

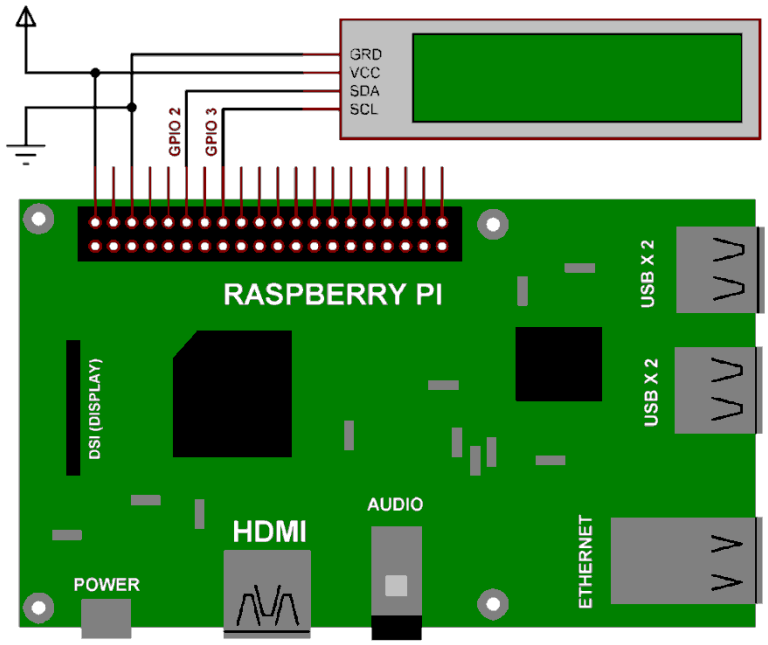
BLOCK DIAGRAM



FLOW CHART


START
Call I2C LCD header file
Initialize LCD
NO
STOP
if Infinite
Loop True
YES
Print on LCD as
HELLO WORLD
*****
Delay 2 Second
Clear LCD
Delay 2 Second

CIRCUIT DIAGRAM



TERMINAL COMMAND LINE

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo raspi-config
4. Select Interfacing Options > I2C.
5. Select Yes when prompted to enable the I2C interface.
6. Select Yes when prompted to automatically load the I2C kernel module.
7. Select Finish.
8. i2cdetect -y 1

PROCEDURE:

1. Start **Thonny** by clicking on the **Raspberry Pi icon** followed by **Programming > Thonny Python IDE.**
2. Write your program in the top pane, click **File > Save** as... to save it,
3. Make hardware connections
4. Click **Run** > Run current script to execute the program. Output will appear in the bottom interpreter pane.

Result

Interface DHT11 Sensor and I2C LCD with Raspberry Pi

AIM:

To interface the DHT11 Sensor and I2C LCD with Raspberry Pi using Thoony Python IDE

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Raspberry Pi Unit • I2C LCD Unit • DHT 11 Sensor Unit	1
3	Jumper Wires	AS Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Call I2C LCD Library and DHT11 Sensor Library
3. Define GPIO Pins
4. Variable Declaration
5. Initiate LCD
6. if Infinite loop True:

    Read Temperature and Humidity Value

    Clear LCD

    Set LCD Cursor (0,0) Position 0 and Column 1

    Print **TEMP: VALUE**

    Set LCD Cursor (0,1) Position 0 and Column 2

Print **HUMI: VALUE**

Delay 3 Second

5. else

Stop Program

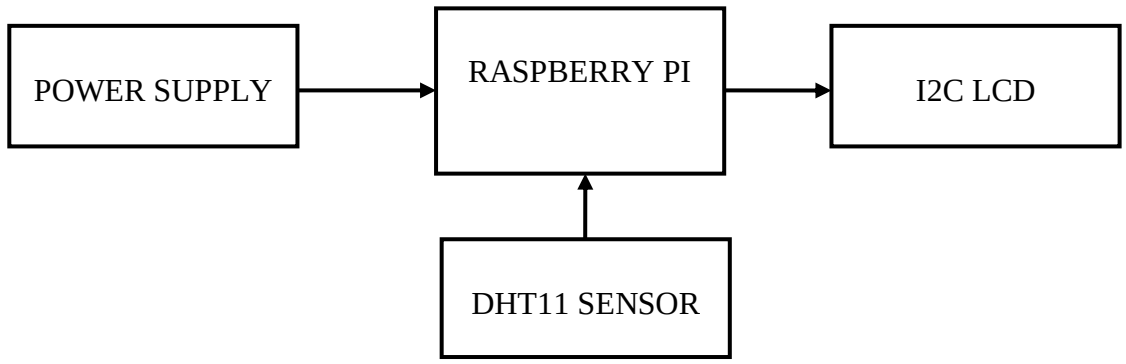
PROGRAM

PROGRAM	COMMENT
import lcddriver	Import LCD Library
from time import *	Import Delay Library
lcd = lcddriver.lcd()	Initialize LCD
import Adafruit_DHT	Import DHT Library
import time	Import Time Library
DHT_SENSOR = Adafruit_DHT.DHT11	Define DHT MOdel DHT11
DHT_PIN = 25	DHT11 Data pin Connected with GPIO 25
while True:	Infinite Loop Starts
humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)	Read Temperature And Humidity Value
if humidity is not None and temperature is not None:	Validate the Read Values
print("Temp={0:0.1f}C Humidity={1:0.1f}%".format(temperature, humidity))	Print Temperature and Humidity Values on Console
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string("Temp: %d%s C" % (temperature, chr(223)), 1)	Print Temperature Value on LCD's 1st Line
lcd.lcd_display_string("Humidity: %d %%" % humidity, 2)	Print Humidity Value on LCD's 2nd Line
else:	Non _ Validated Values
print("DATA READING....");	Reading Again
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string("DATA READING...",1)	Print **DATA READING...** on LCD's 1st Line
lcd.lcd_display_string(" ****" , 2)	Print ******** on LCD's 2nd Line
time.sleep(3);	Delay 3 Second

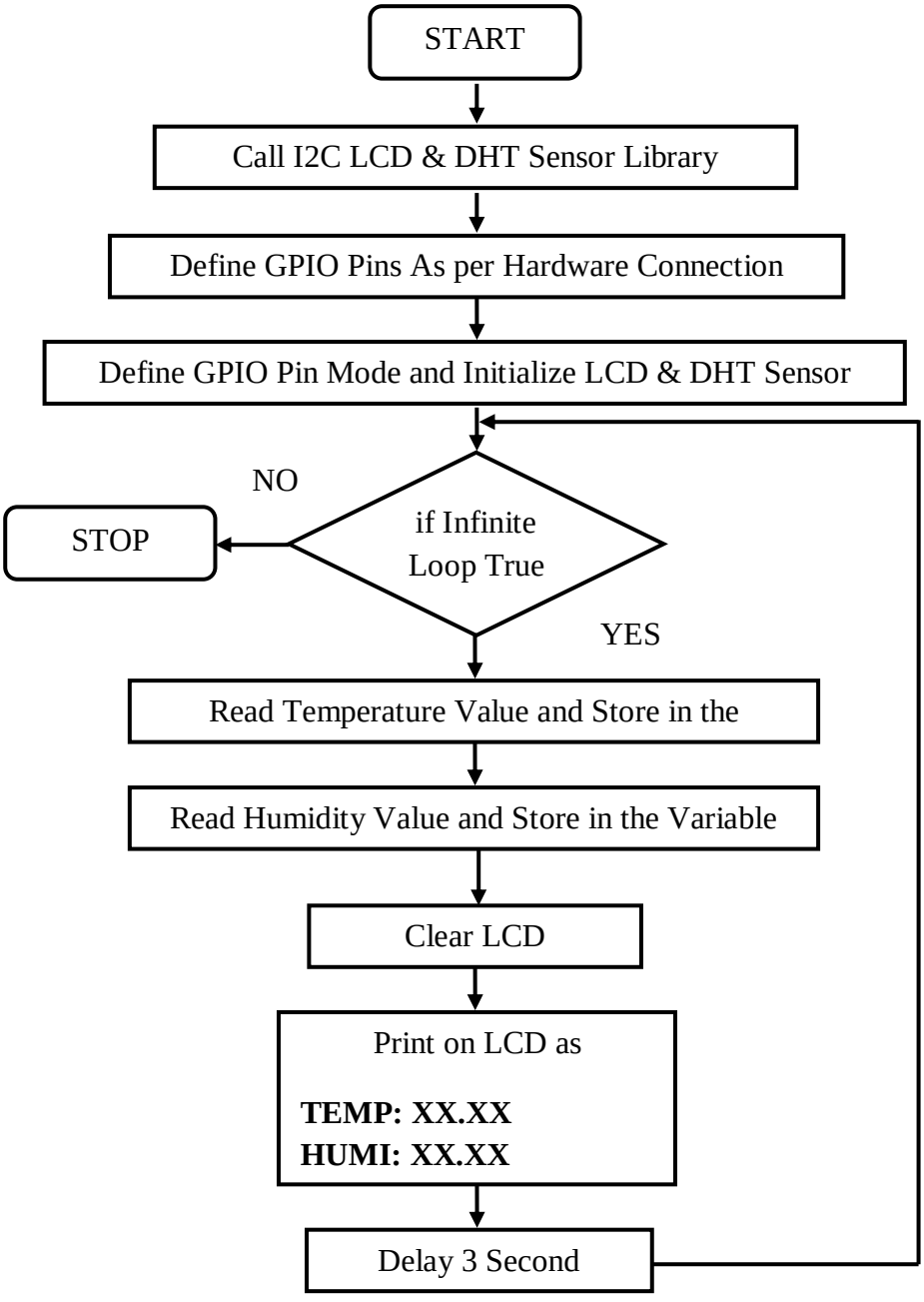
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	RASPBERRY PI	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LCD (SDA)	SPIO 02	Serial Data
4	LCD (SCL)	GPIO 03	Serial Clock
5	DHT11	GPIO 25	Digital Input

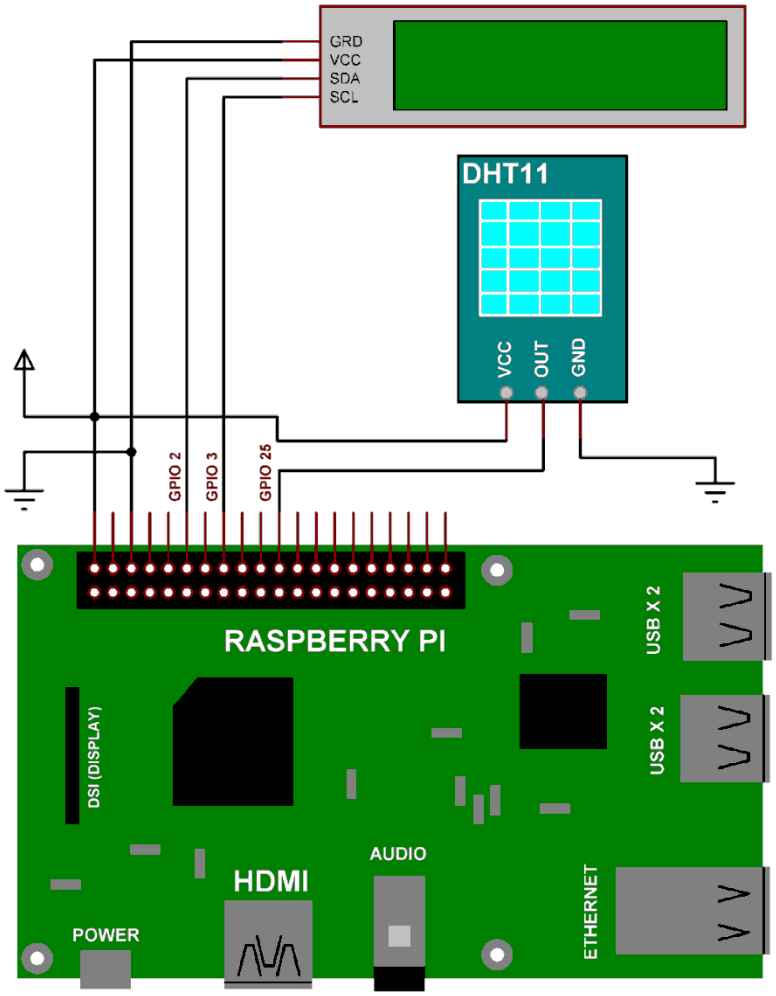
BLOCK DIAGRAM



FLOW CHART


START
Call I2C LCD & DHT Sensor Library
Define GPIO Pins As per Hardware Connection
Define GPIO Pin Mode and Initialize LCD & DHT Sensor
NO
STOP
if Infinite
Loop True
YES
Read Temperature Value and Store in the
Read Humidity Value and Store in the Variable
Clear LCD
Print on LCD as
TEMP: XX.XX
HUMI: XX.XX
Delay 3 Second

CIRCUIT DIAGRAM



TERMINAL COMMAND LINE

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo raspi-config
4. Select Interfacing Options > I2C.
5. Select Yes when prompted to enable the I2C interface.
6. Select Yes when prompted to automatically load the I2C kernel module.
7. Select Finish.
8. i2cdetect -y 1
9. sudo raspi-config
10. Select Interfacing Options >1 WIRE.

11. Select Yes when prompted to enable the 1 WIRE interface.
12. Select Yes when prompted to automatically load the 1 WIRE kernel module.
13. Select Finish.

PROCEDURE:

1. Start **Thonny** by clicking on the **Raspberry Pi icon** followed by **Programming > Thonny Python IDE.**
2. Write your program in the top pane, click **File > Save** as... to save it,
3. Make hardware connections
4. Click **Run** > Run current script to execute the program. Output will appear in the bottom interpreter pane.

Result

Communicate between Arduino and Raspberry PI using I2C Protocol

AIM:

To **:** Interface and Communication between Raspberry pi and Arduino Nano Using I2C Protocol

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Raspberry Pi Unit • Arduino Nano Unit • LED Array Unit	1
3	Jumper Wires	As Required
4	USB Cable	1

RASPBERRY PI ALGORITHM:

1. Start the program
2. Define I2C Bus and Address
3. if Infinite loop True:

   Read Data from Interpreter Pane

   if Data ==1

   Send 0x1 to Arduino

   else if Data==0

   Send 0x0 to Arduino
4. else

   Stop Program

ARDUINO ALGORITHM:

1. Start the program
2. Define I2C Bus and Address
3. if Infinite loop True:

    Read Data from I2C port

    if Data ==1

    LED ON

    if Data==0

    LED OFF

6. else

    Stop Program

RASPBERRY PI PROGRAM

PROGRAM	COMMENT
from smbus import SMBus	Import I2C Bus
addr = 0x8	Define Address For Bus
bus = SMBus(1)	Initialize I2C Bus
print ("Enter 1 for ON or 0 for OFF")	Print ON/OFF Instruction on Console
while numb == 1:	Infinite Loop Starts
ledstate = input(">>>> ")	Read Console Data
if ledstate == "1":	Check ON State
bus.write_byte(addr, 0x1)	Send 1 to Arduino
elif ledstate == "0":	Check OFF State
bus.write_byte(addr, 0x0)	Send 0 to Arduino
else:	Wait For Data

ARDUINO PROGRAM

PROGRAM	COMMENT
#include <Wire.h>	Call 1 Wire Library
const int ledPin = 2;	LED connected with D2
void setup() {	Set up Loop Start
Wire.begin(0x8);	Define I2C Address
Wire.onReceive(receiveEvent);	Define I2C Interrupt Function
pinMode(ledPin, OUTPUT);	LED Configures as OUTPUT
digitalWrite(ledPin, LOW);	LED OFF
}	Set up Loop Close
void receiveEvent(int howMany) {	I2C Interrupt Function Calling
while (Wire.available()) {	Check I2C Data
char c = Wire.read();	Read I2C Data and Stored in Variable
digitalWrite(ledPin, c);	Toggle LED as per I2C Data
} }	I2C Interrupt Function Close

void loop() {
delay(100);
}
	Infinite Loop

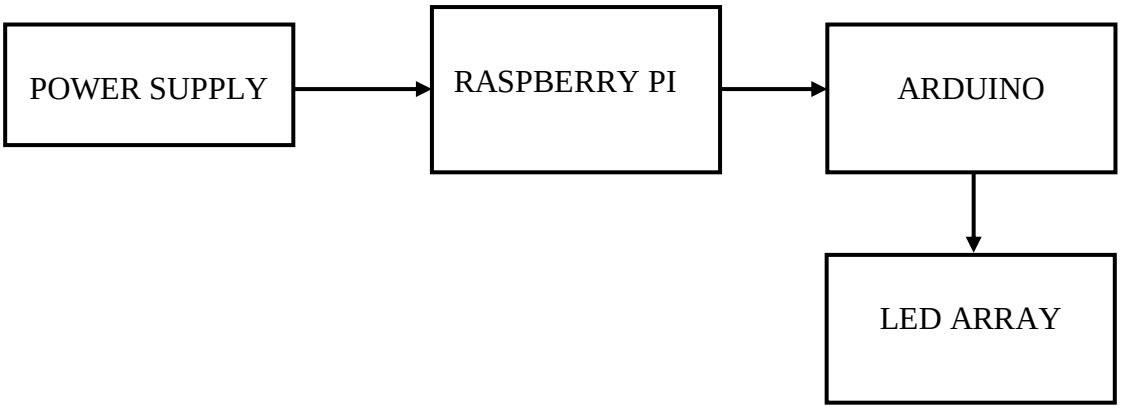
RASPBERRY PI CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	RASPBERRY PI	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	ARDUINO SDA (A4)	SDA (GPIO 2)	I2C DATA
4	ARDUINO SCL (A5)	SCL (GPIO 3)	I2C CLOCK

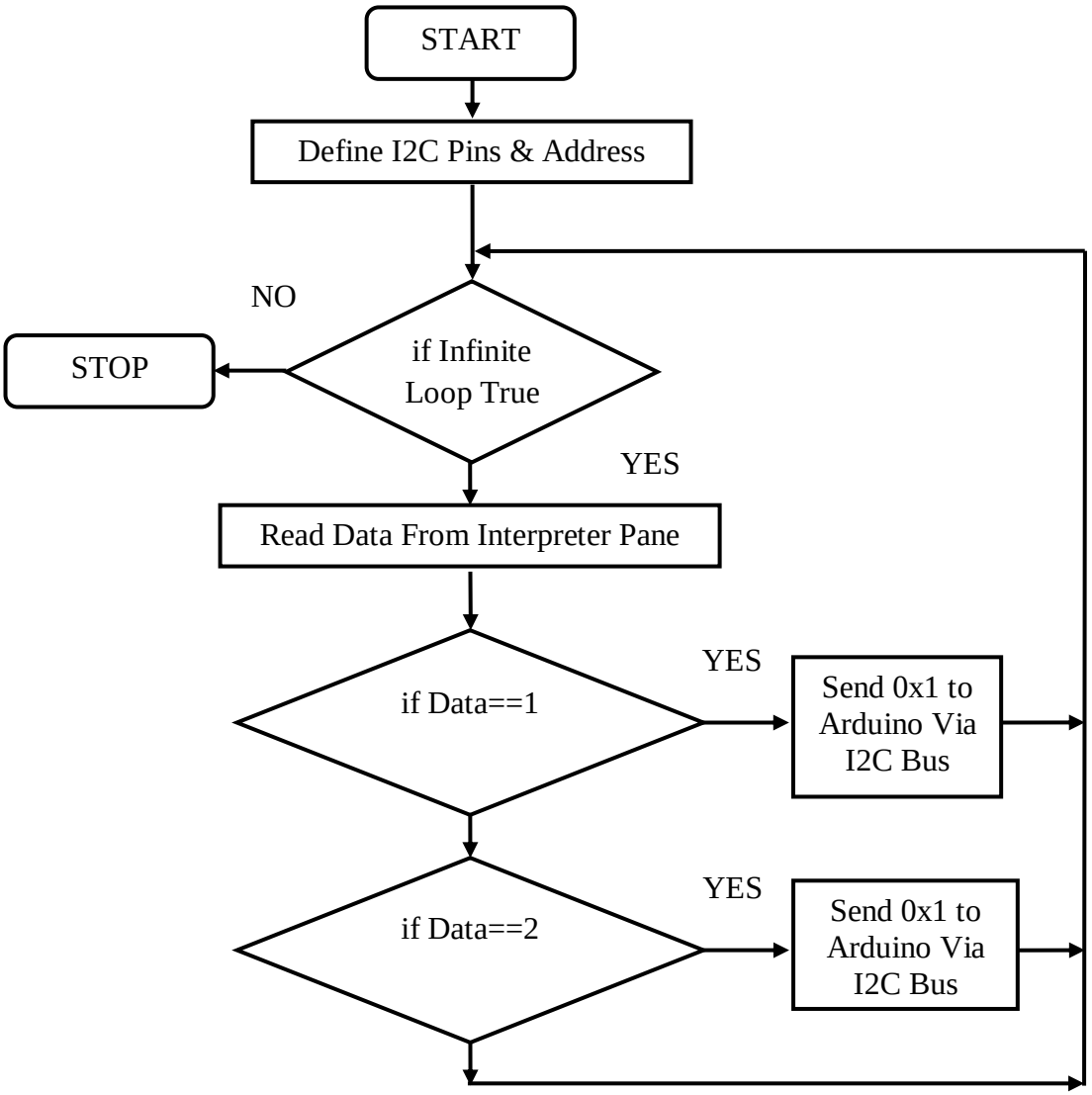
RASPBERRY PI CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	ARDUINO	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LED1	D2	OUTPUT

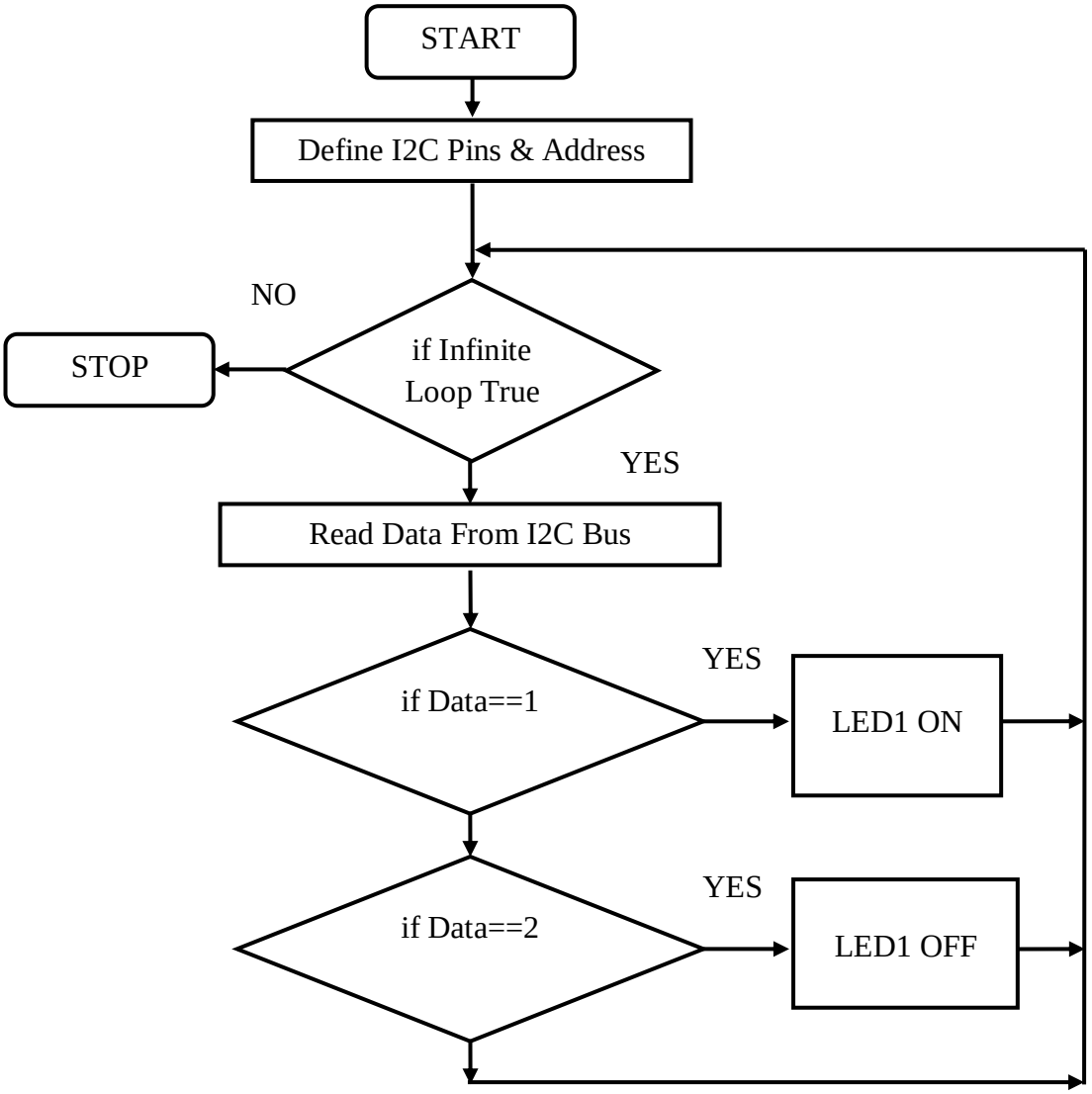
BLOCK DIAGRAM



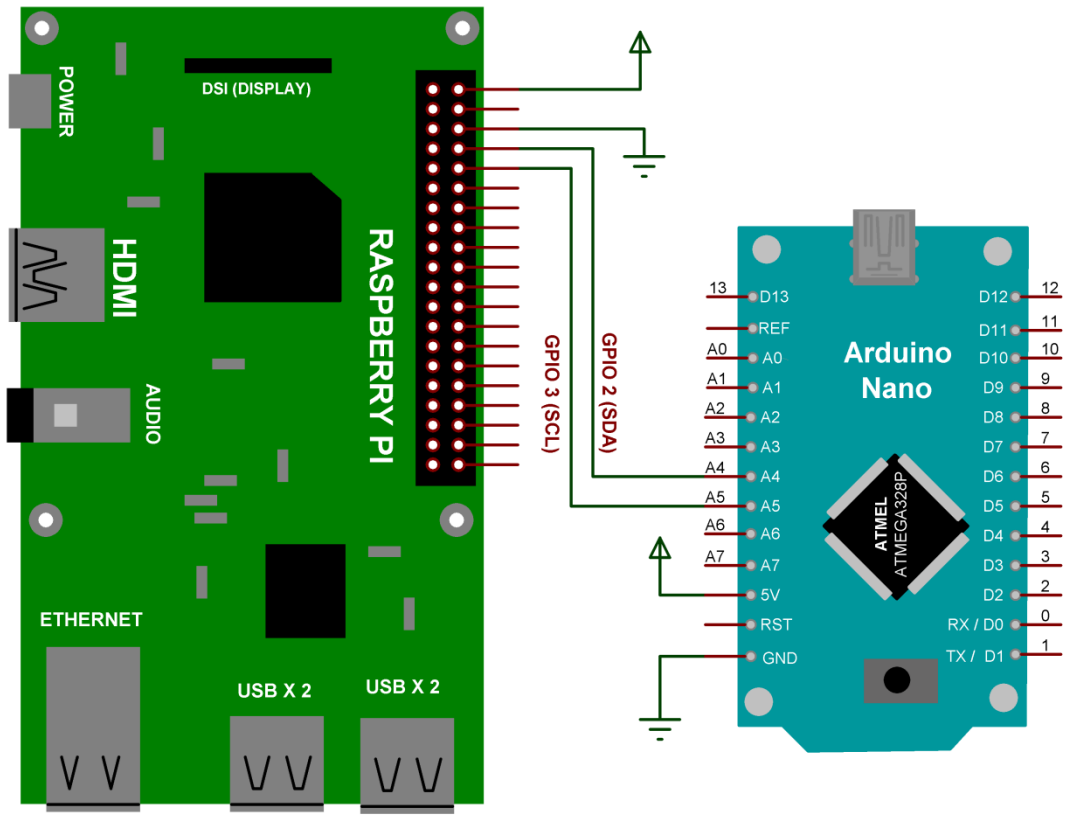
RASPBERRY PI FLOW CHART



ARDUINO FLOW CHART



CIRCUIT DIAGRAM



TERMINAL COMMAND LINE

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo raspi-config
4. Select Interfacing Options > I2C.
5. Select Yes when prompted to enable the I2C interface.
6. Select Yes when prompted to automatically load the I2C kernel module.
7. Select Finish.
8. i2cdetect -y 1
9. sudo raspi-config
10. Select Interfacing Options >1 WIRE.
11. Select Yes when prompted to enable the 1 WIRE interface.
12. Select Yes when prompted to automatically load the 1 WIRE kernel module.
13. Select Finish.

PROCEDURE:

1. Start **Thonny** by clicking on the **Raspberry Pi icon** followed by **Programming > Thonny Python IDE.**
2. Write your program in the top pane, click **File > Save** as... to save it,
3. Make hardware connections
4. Click **Run** > Run current script to execute the program. Output will appear in the bottom interpreter pane.

PROCEDURE:

1. Open **ARDUINO IDE** Software
2. Click **File >> New** or **CTRL+N** to create New Project.
3. To Select Board Click **Tools >> Board > Arduino AVR Boards >> Arduino Nano**
4. To select Processor Click **Tools >> Processor >> ATmega328P (Old Bootloader)**
5. To select Programmer Click **Tools >> Programmer >> AVRISP mkII**
6. Click **File >> Save** or **CTRL+S** to Save Project where we Desire.
7. Type the Program.
8. Click **Sketch >> Verify / Compile** or **CTRL+R** to compile the program (If any error present debug the program)
9. Connect the Arduino Nano with PC / Laptop by USB Cable
10. Select the COM PORT by Click **Tools >> Port** and Select Arduino's COM PORT Number.

Note : If COM PORT Number is Don't Know:

- Right Click My Computer Icon and Select Properties
- Click Device Manager
- Expand Ports (COM & LPT). COM PORT Number will be shown

11. Click **Sketch >> Upload** or **CTRL+U** to upload the program to Arduino Nano.
12. Switch On the corresponding Units we need and Verify the Program.

Result

Create New Project And Setup A Firebase Cloud Platform To Log The Data

.

Setup a cloud platform to log the data

AIM:

To Create New Project and Setup Firebase Real-time Database.

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse With High Speed Internet Facility	1

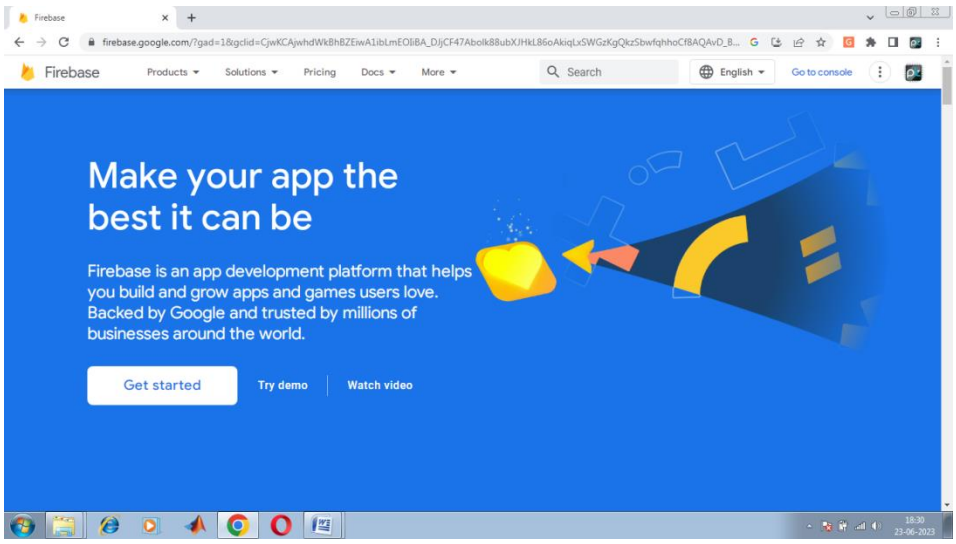
STEP_1:

- First, Visit Firebase Console Using This Visiting The Following URL - https://console.firebase.google.com.

SPEP_2:

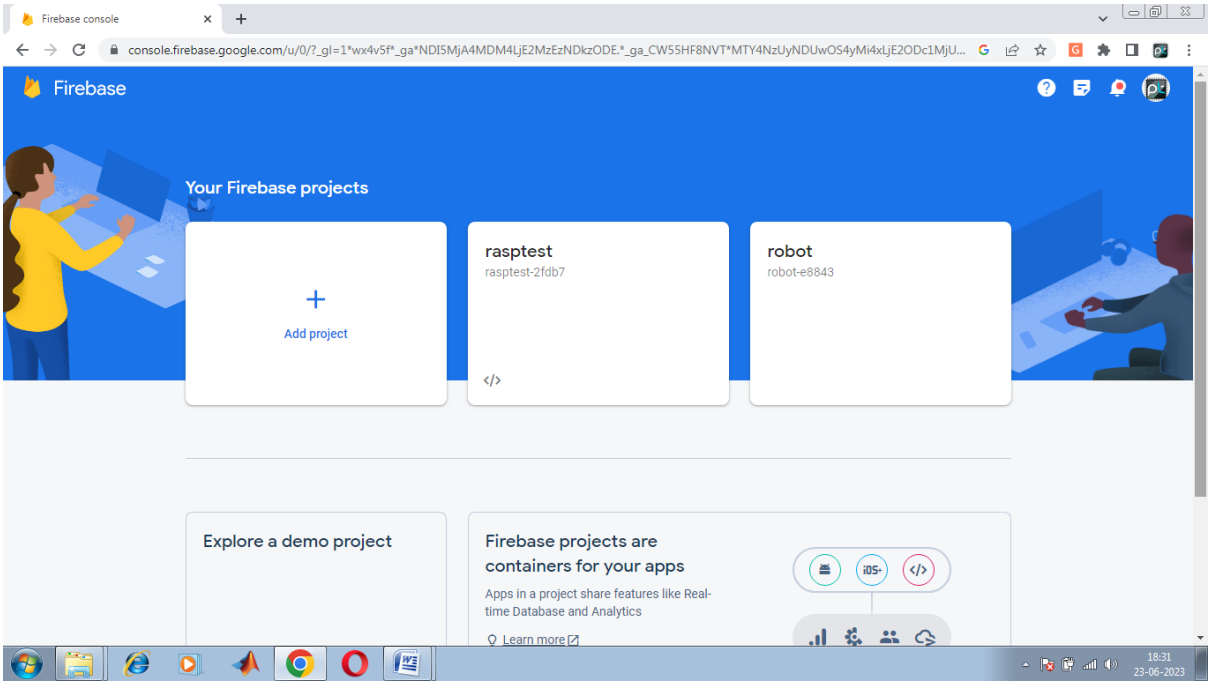
- Login Using Your Google Account - If You Are Not Already Logged In.

STEP_3:



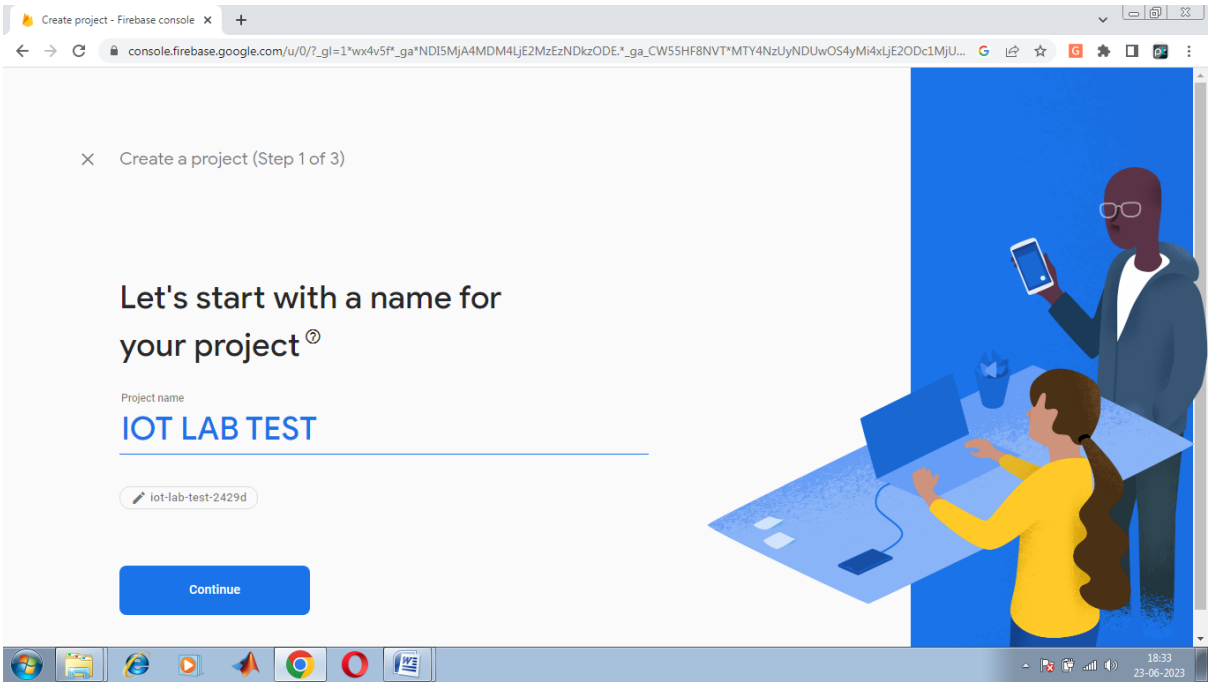
- Click **Get Started**

STEP_4:



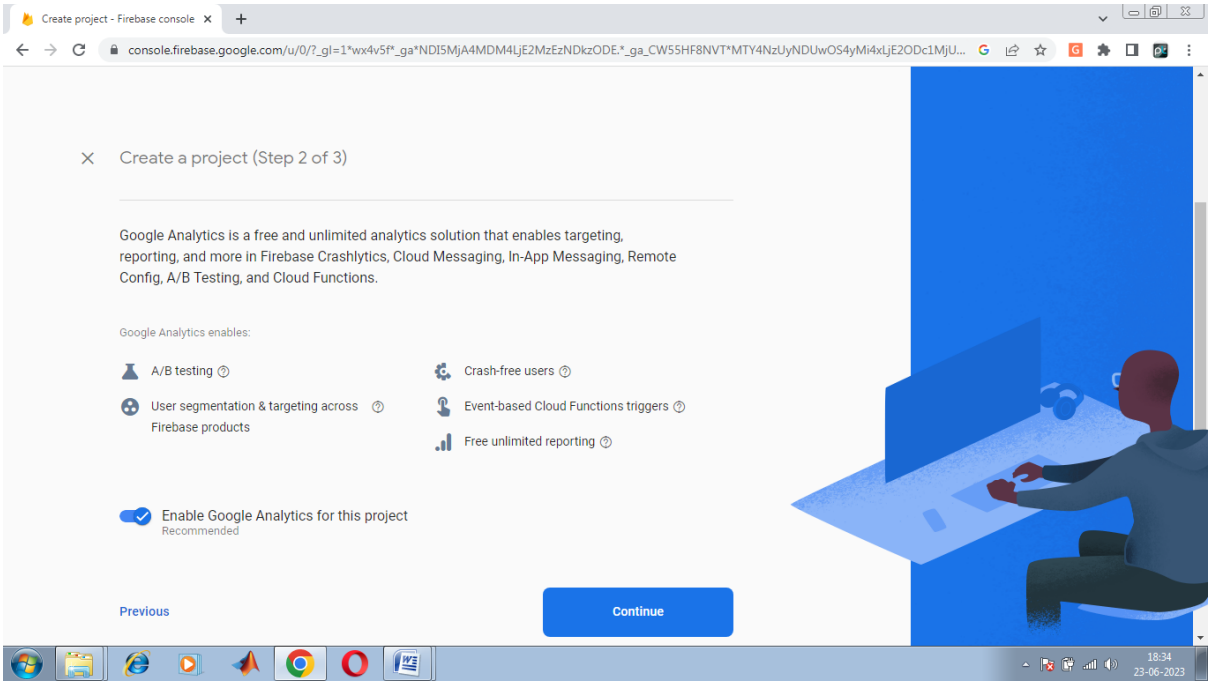
- Click **Add Prjoject**

STEP_5:



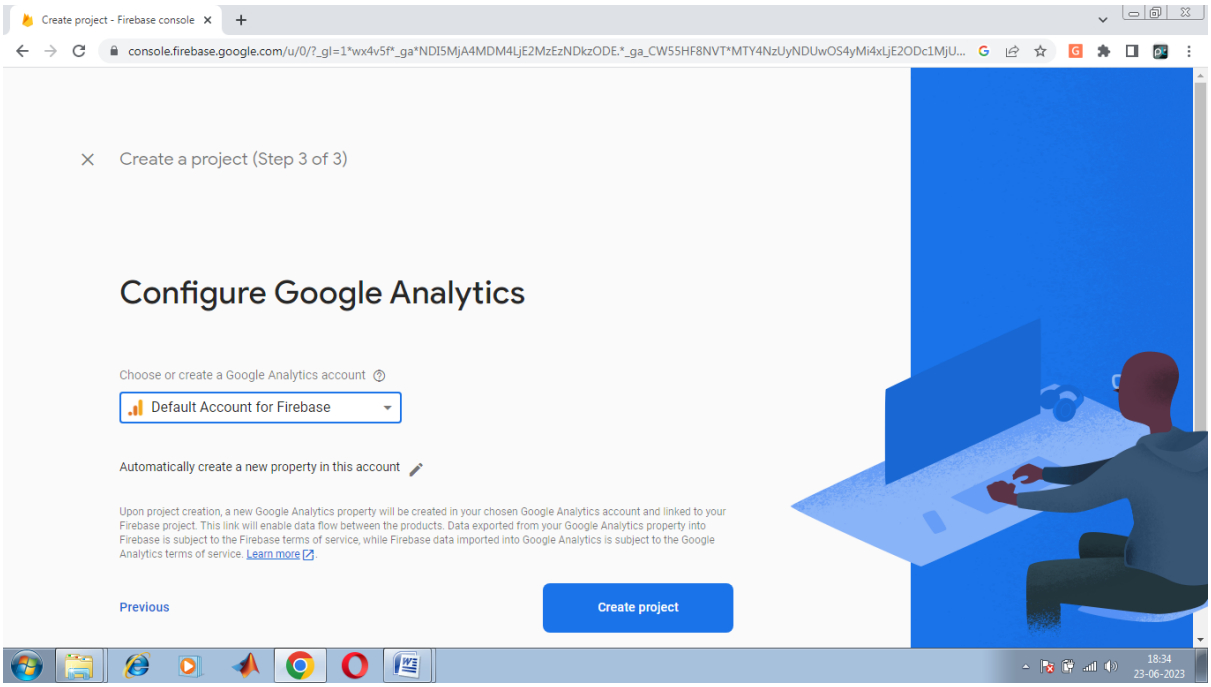
- Enter Your **Project Name and** Click **Continue**

STEP_6:



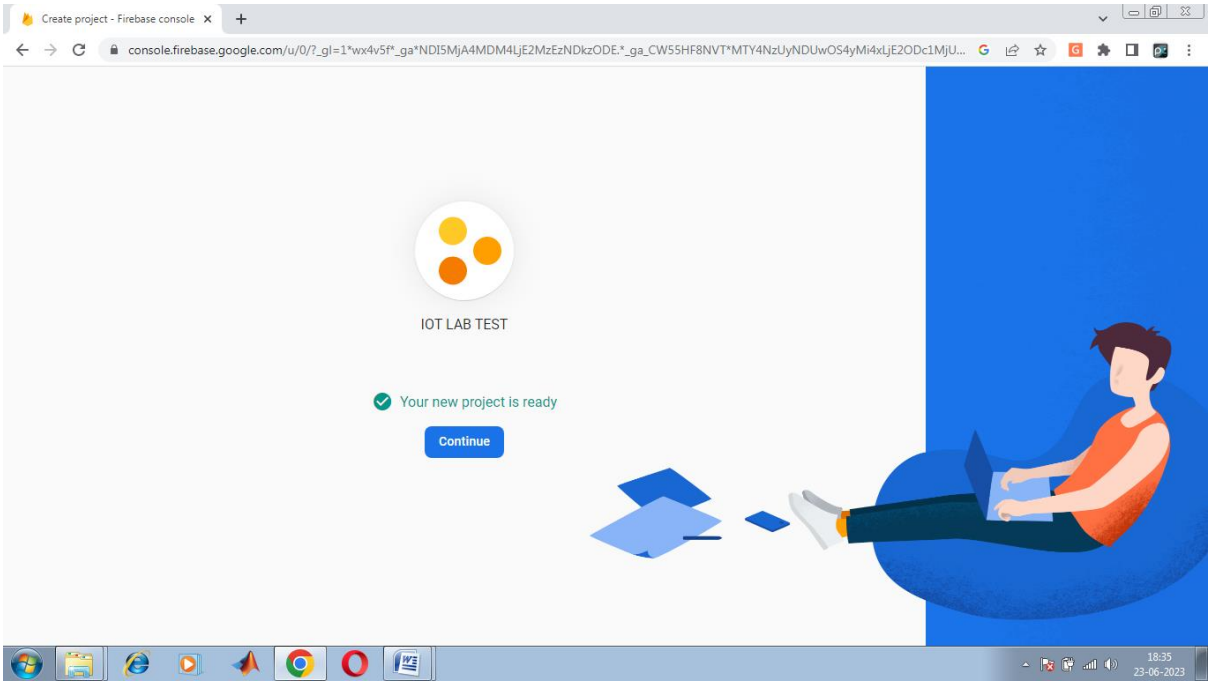
- Click **Continue**

STEP_7:



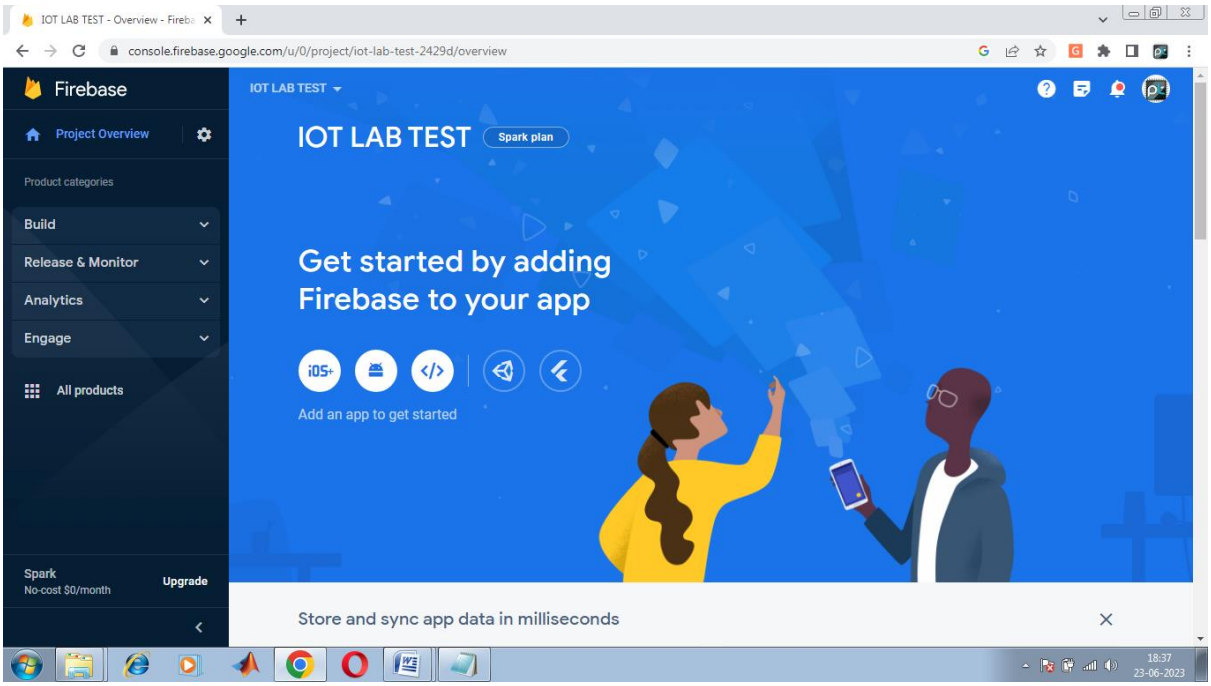
- Select **Default Account For Firebase** And Click **Create Project**

STEP_8:



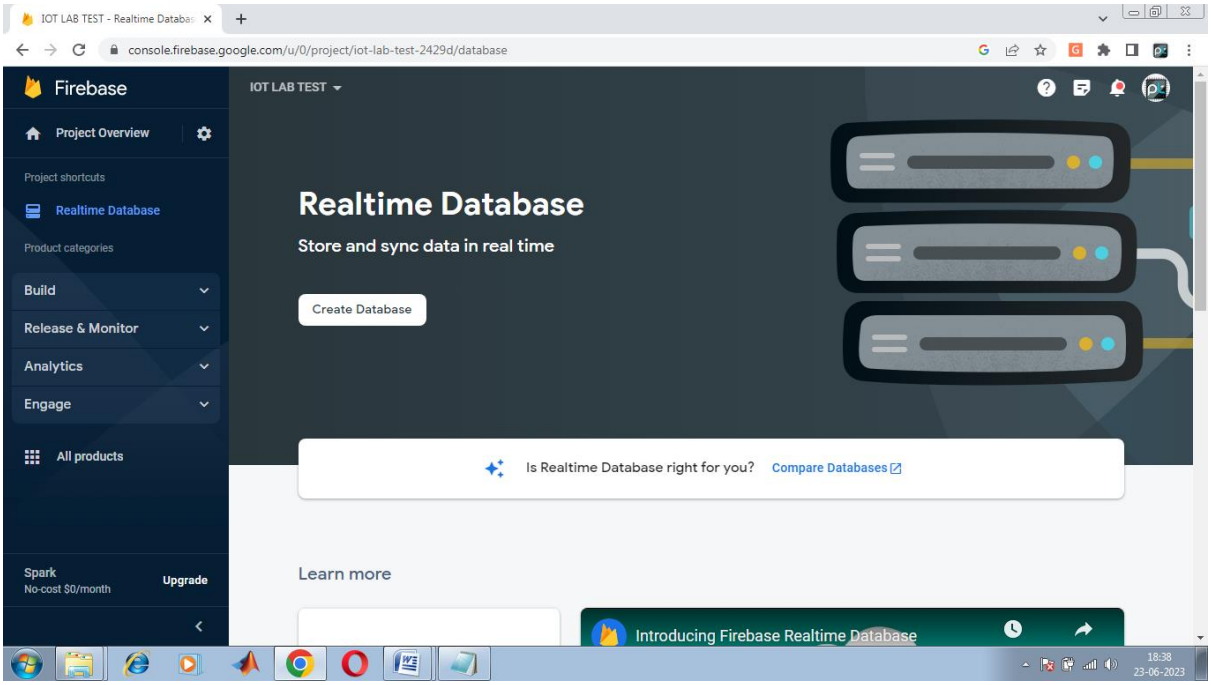
- Project Is Created And Click **Continue**

STEP_9:



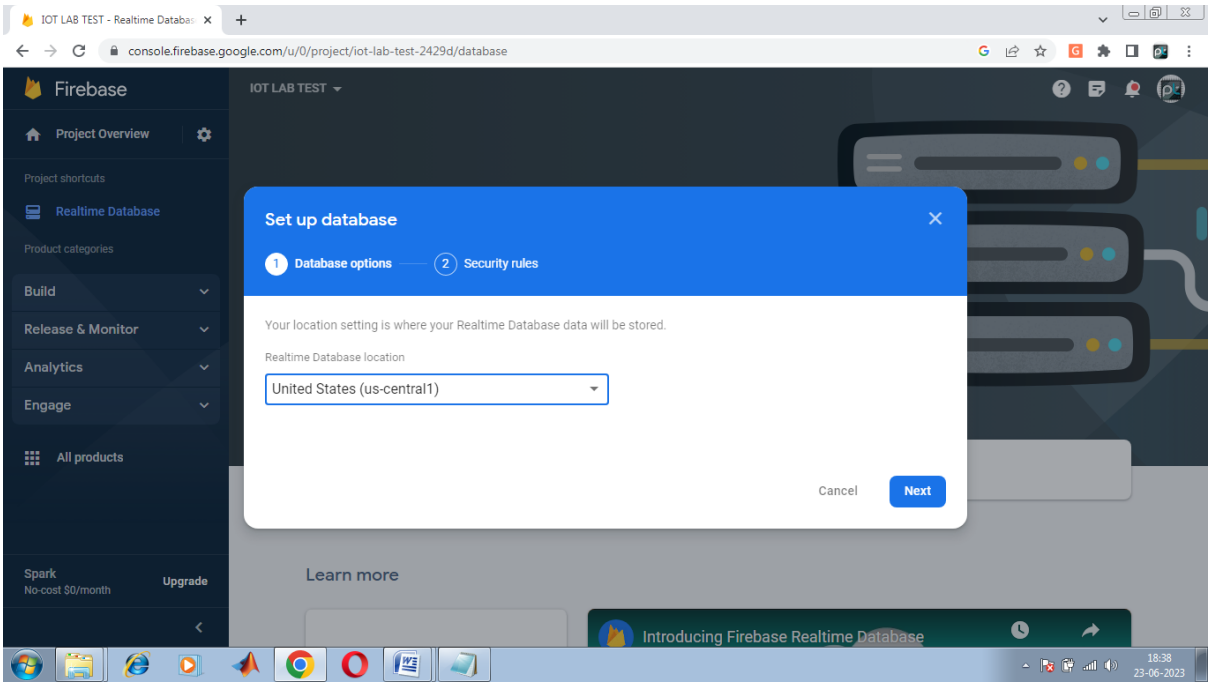
- Firebase Console Home Page Opened and Click **Build And Select Real Time Database**

STEP_10:



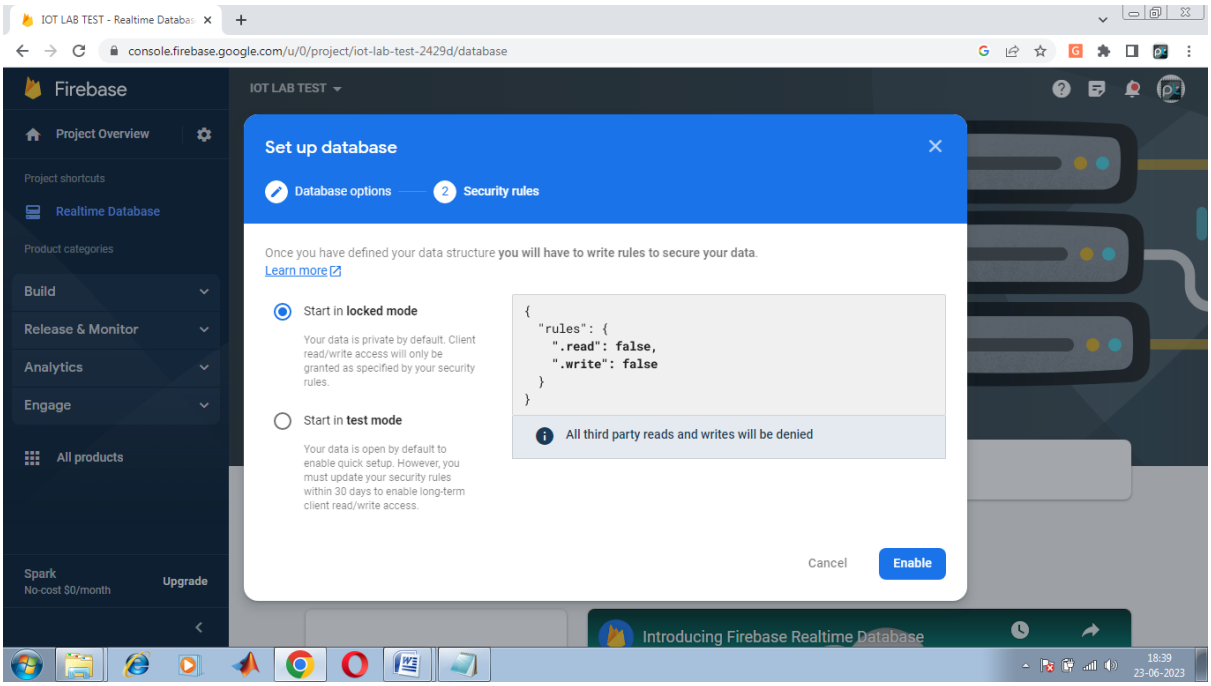
- Click **Create Database**

STEP_11:



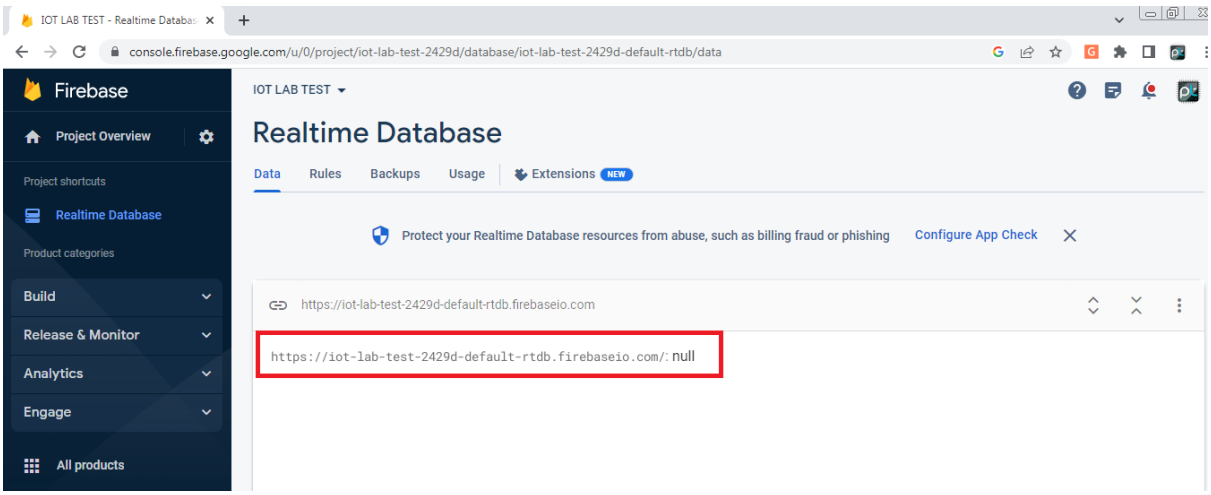
- Select **Realtime Database Location** As **United States (Us-Central1)** And Click **Next**

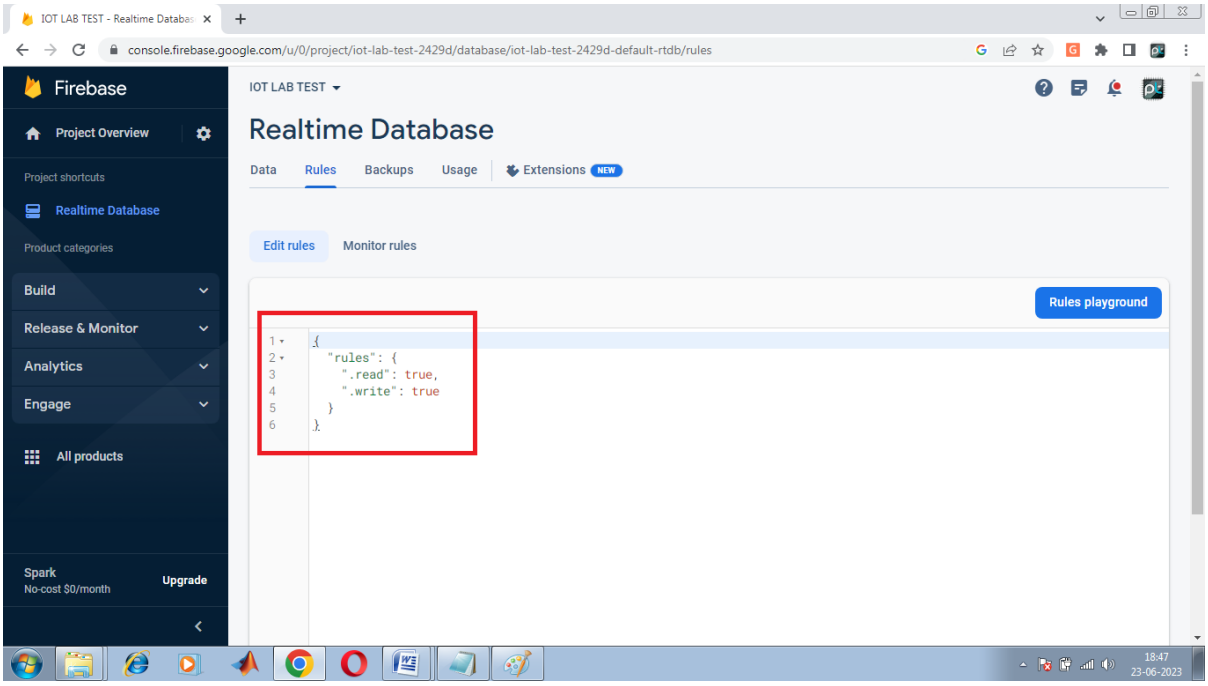
STEP_12:



- Select **Start In Locked Mode** And Click **Enable**

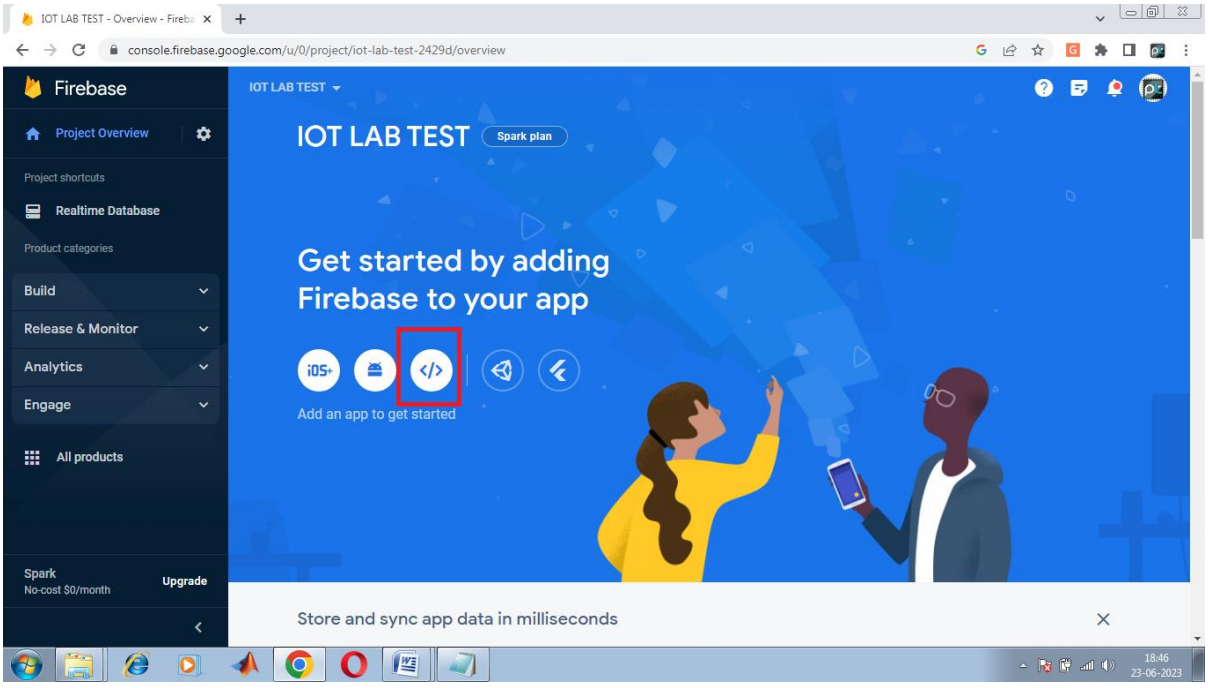
STEP_13:





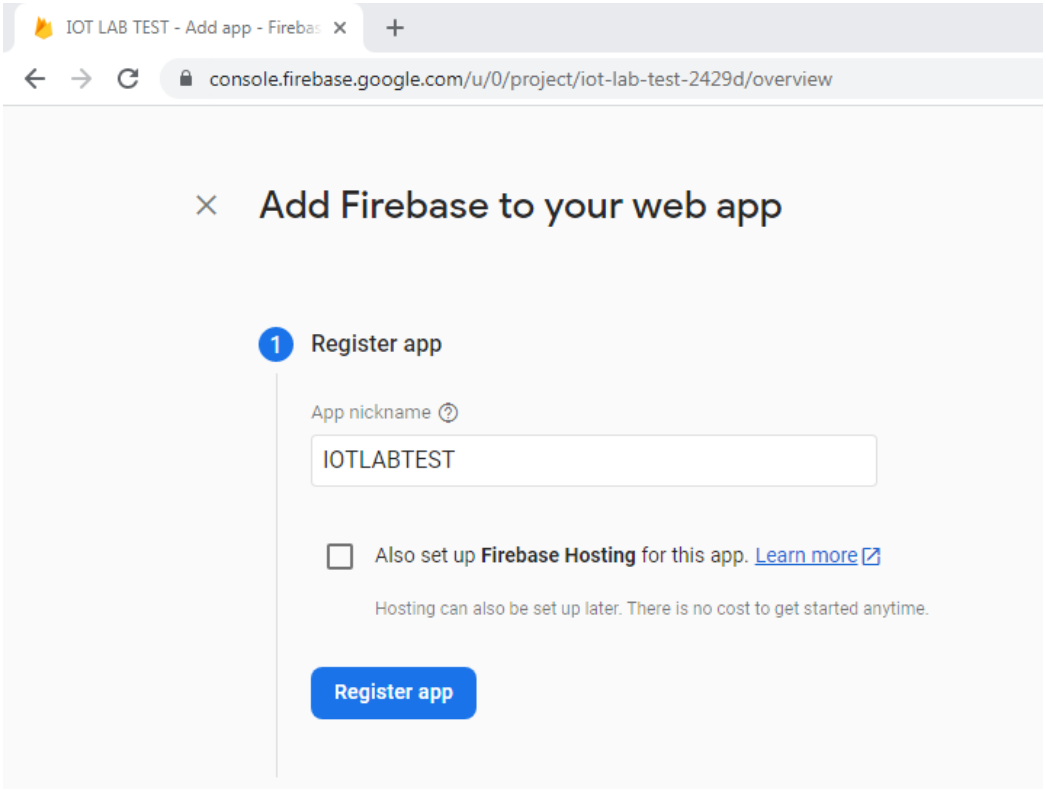
- Now **Real Time Database Is Created** With Host URL.Click Rules And Change **Write And Read Rules As true** And Click **Publish**

STEP_14:



- Click **Project Overview** . Click **Web Link** Mentioned In Red Box

STEP_15:



- Enter **App Name** And Click **Register App**

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyBU-jPZB6aLXoAIt--vQPI8Frf1DV6PR0Q",
  authDomain: "iot-lab-test-2429d.firebaseapp.com",
  databaseURL: "https://iot-lab-test-2429d-default-rtdb.firebaseio.com",
  projectId: "iot-lab-test-2429d",
  storageBucket: "iot-lab-test-2429d.appspot.com",
  messagingSenderId: "1095415346205",
  appId: "1:1095415346205:web:1e861499d1ee8a56ce42f4",
  measurementId: "G-KLJ4EZHJQ9"
};
```

- Copy These Line Saved In Notepad File For Future IOT Project Deployment
- Firebase Configuration Data's Are Created. Click **Continue to Console**

Result

Design and Implementation of IOT Based Weather Station and Log Humidity Sensor Data using Raspberry PI and upload to the cloud platform

AIM:

To Design and Implement IOT Based Weather Station using Raspberry Pi.

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Raspberry Pi Unit • I2C LCD Unit • DHT11Sensor Unit	1
3	Jumper Wires	AS Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Call GPIO, DHT11, FIREBASE And I2C LCD Library
3. Initialize LCD, DHT11 And FIREBASE
4. Variable Declaration
5. Define GPIO Pins And Its Mode
6. if Infinite loop True:

    Read Temperature And Humidity Value

    Clear LCD

    Print **TEMP: VALUE**

Print **HUMI: VALUE**

Update Temperature And Humidity Value on IoT Firebase Database

if Temperature >=40

Print **CLIMATE**

Print **SUNNY**

Update Climate Status on IoT Firebase Database

else if humidity >=80

Print **CLIMATE**

Print **RAINY**

Update Climate Status on IoT Firebase Database

else

Print **CLIMATE**

Print **NORMAL**

Update Climate Status on IoT Firebase Database

7. else

Stop Program

PROGRAM

PROGRAM	**COMMENT**
import lcddriver	Import LCD Library
from time import *	Import Delay Library
lcd = lcddriver.lcd()	Initialize LCD
import RPi.GPIO as GPIO	import GPIO library
import pyrebase	Import Firebase Library
from time import sleep	Import Delay Library
import time	Import Time Library
import Adafruit_DHT	Import DHT Library
import time	Import Time Library
DHT_SENSOR = Adafruit_DHT.DHT11	Define DHT Model as DHT11
DHT_PIN = 25	DHT11 Data Pin Connected With GPIO 25
config = { "apiKey": "wgsNjBl7qvzVY0KiezCBplSlRxLI9OiPwEschMfh", "authDomain": "rasptest-	configure firebase database Variables

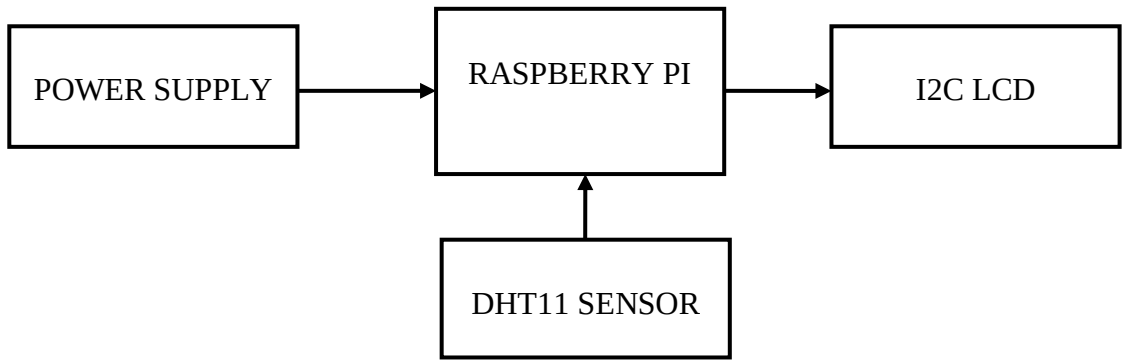
2fdb7.firebaseapp.com", "databaseURL": "https://rasptest-2fdb7-default-rtdb.firebaseio.com/", "storageBucket": "rasptest123" }	
firebase = pyrebase.initialize_app(config)	Initialize Firebase
GPIO.setmode(GPIO.BCM)	Define as Physical numbering
GPIO.setwarnings(False)	Warning Off for GPIO pins
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" IoT Based ", 1)	Print **IoT Based** on LCD 1st Line
lcd.lcd_display_string("Weather Station" , 2)	Print **Weather Station** on LCD 2nd Line
time.sleep(2)	Delay 2 Second
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" System", 1)	Print **System** on LCD 1st Line
lcd.lcd_display_string(" *******" , 2)	Print ************ on LCD 2nd Line
time.sleep(2)	Delay 2 Second
try:	Test a block of code for errors
while True:	Infinite Loop Start
database = firebase.database()	Read Firebase Database
ProjectBucket = database.child("IOTLAB")	Define Firebase Data Storage Bucket
LEDDATA = ProjectBucket.child("LED").get().val()	Read Firebase Data
humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)	Read Temperature And Humidity Value
if humidity is not None and temperature is not None:	Validate the Read Values
print("Temp={0:0.1f}C Humidity={1:0.1f}%".format(temperature, humidity))	Print Temperature and Humidity Values on Console
ProjectBucket.child("IOTLAB").child("Humidity").set(humidity)	Update Humidity Value on IoT Firebase Database
ProjectBucket.child("IOTLAB").child("Temperature").set(temperature)	Update Temperature Value on IoT Firebase Database
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string("Temp: %d%s C" % (temperature, chr(223)), 1)	Print Temperature Value on LCD's 1st Line
lcd.lcd_display_string("Humidity: %d %%" % humidity, 2)	Print Humidity Value on LCD's 2nd Line
else:	Non _ Validated Values

print("DATA READING....");	Reading Again
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string("DATA READING...",1)	Print **DATA READING...** on LCD's 1st Line
lcd.lcd_display_string(" ****" , 2)	Print ******** on LCD's 2nd Line
time.sleep(2)	Delay 2 Second
if temperature >=40:	Check SUNNY Loop
print("SUNNY")	Print **SUNNY** on Console
ProjectBucket.child("IOTLAB").child("Climate").set("Sunny")	Update CLIMATE as SUNNY on IoT Firebase Database
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" CLIMATE",1)	Print **CLIMATE.** on LCD's 1st Line
lcd.lcd_display_string(" SUNNY" , 2)	Print **SUNNY** on LCD's 2nd Line
elif humidity>=80:	Check RAINY Loop
print("RAINY")	Print **RAINY** on Console
ProjectBucket.child("IOTLAB").child("Climate").set("Rainy")	Update CLIMATE as RAINY on IoT Firebase Database
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" CLIMATE",1)	Print **CLIMATE.** on LCD's 1st Line
lcd.lcd_display_string(" RAINY" , 2)	Print **RAINY** on LCD's 2nd Line
else:	Check NORMAL Loop
print("NORMAL")	Print **NORMAL** on Console
ProjectBucket.child("IOTLAB").child("Climate").set("Normal")	Update CLIMATE as NORMAL on IoT Firebase Database
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" CLIMATE",1)	Print **CLIMATE.** on LCD's 1st Line
lcd.lcd_display_string(" NORMAL" , 2)	Print **NORMAL** on LCD's 2nd Line
time.sleep(2)	Delay 2 Second
except KeyboardInterrupt:	Program Is Interrupted By The User Keyboard
GPIO.cleanup()	Reset GPIO Pins

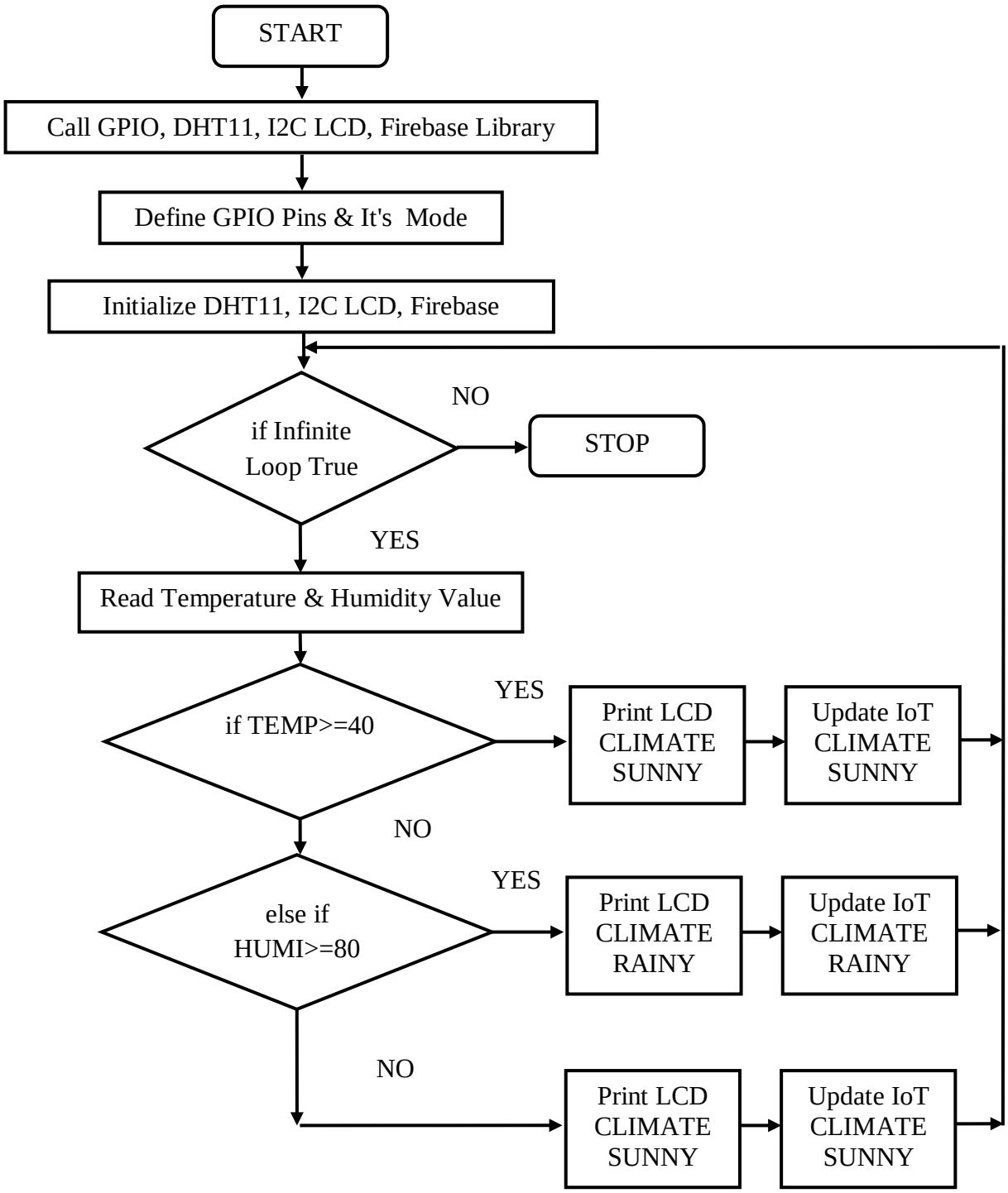
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	RASPBERRY PI	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LCD (SDA)	GPIO 2	Serial Data
4	LCD (SCL)	GPIO 3	Serial Clock
5	DHT11	GPIO 25	Digital Input

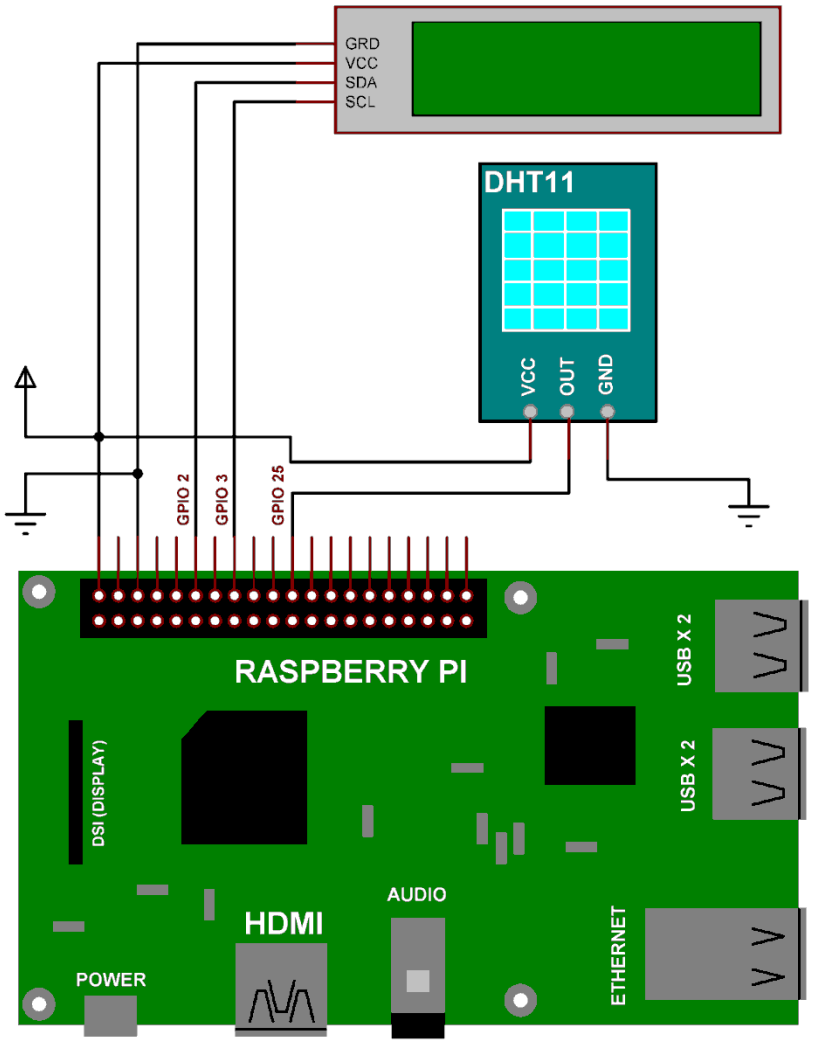
BLOCK DIAGRAM



FLOW CHART


START
Call GPIO, DHT11, I2C LCD, Firebase Library
Define GPIO Pins & It's Mode
Initialize DHT11, I2C LCD, Firebase
if Infinite Loop True
NO
STOP
YES
Read Temperature & Humidity Value
if TEMP>=40
YES
Print LCD CLIMATE SUNNY
Update IoT CLIMATE SUNNY
NO
else if HUMI>=80
YES
Print LCD CLIMATE RAINY
Update IoT CLIMATE RAINY
NO
Print LCD CLIMATE SUNNY
Update IoT CLIMATE SUNNY

CIRCUIT DIAGRAM



TERMINAL COMMAND LINE

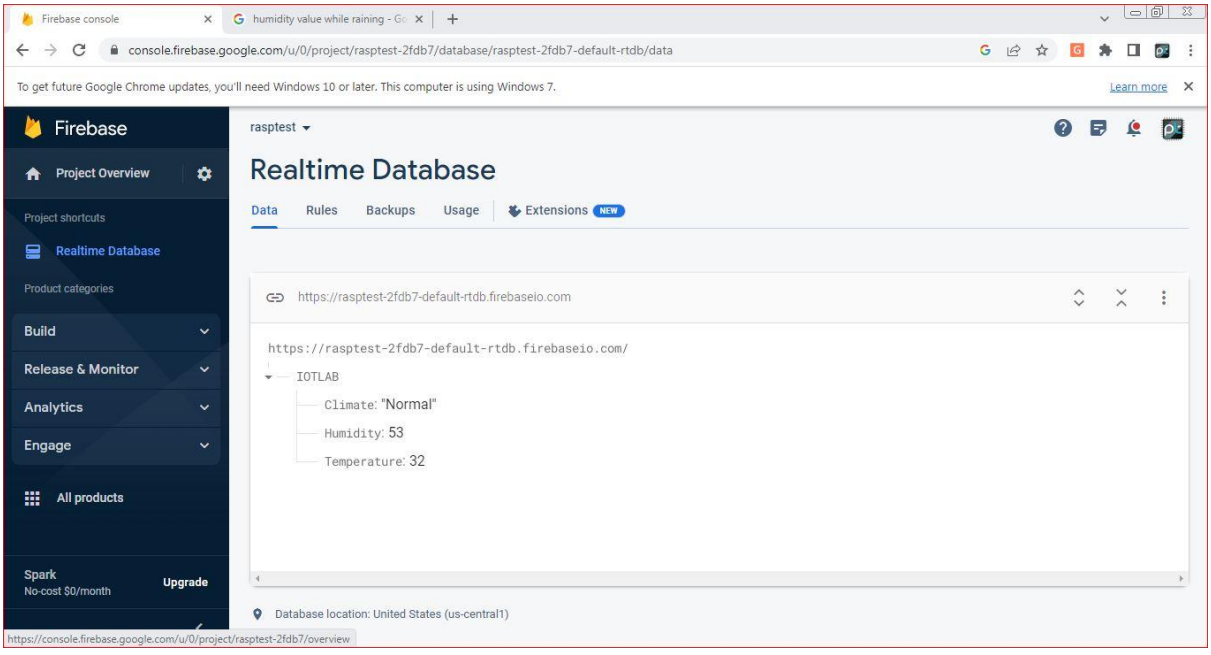
1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo raspi-config
4. Select Interfacing Options > I2C.
5. Select Yes when prompted to enable the I2C interface.
6. Select Yes when prompted to automatically load the I2C kernel module.
7. Select Finish.
8. i2cdetect -y 1

9. sudo raspi-config
10. Select Interfacing Options >1 WIRE.
11. Select Yes when prompted to enable the 1 WIRE interface.
12. Select Yes when prompted to automatically load the 1 WIRE kernel module.
13. Select Finish.
14. sudo apt-get update.
15. sudo apt-get install python-dev.
16. sudo python get-pip OR sudo apt-get install python-pip (new Raspian versions)
17. sudo pip install pyrebase.

PROCEDURE:

1. Start **Thonny** by clicking on the **Raspberry Pi icon** followed by **Programming > Thonny Python IDE.**
2. Write your program in the top pane, click **File > Save** as... to save it,
3. Make hardware connections
4. Click **Run** > Run current script to execute the program. Output will appear in the bottom interpreter pane.

RESULT SCREEN SHOT




https://rasptest-2fdb7-default-rtdb.firebaseio.com
https://rasptest-2fdb7-default-rtdb.firebaseio.com/
IOTLAB
Climate: "Normal"
Humidity: 53
Temperature: 32

Result

Design And Implementation Of IoT Based Home Automation Using Raspberry Pi

AIM:

To Design and Implement IOT Based Home Automation using Raspberry Pi.

REQUIRED ITEMS

S.NO	ITEM	QTY
1	PC / LAPTOP **SYSTEM SPECIFICATION:** • OS : windows 7 or above • Hard disk : 256 GB or above • RAM : 2 GB or above • Keyboard and Mouse	1
2	CS3691 IOT TRAINER KIT • Raspberry Pi Unit • Relay Unit	1
3	Jumper Wires	AS Required
4	USB Cable	1

ALGORITHM:

1. Start the program
2. Call GPIO, FIREBASE And I2C LCD Library
3. Initialize LCD, And FIREBASE
4. Variable Declaration
5. Define GPIO Pins And Its Mode
6. if Infinite loop True:

    Read Firebase Data

    if LED1Data =1

    Relay_1 ON

    else

    Realy_1 OFF

if LED2Data =1

Relay_2 ON

else

Realy_2 OFF

Clear LCD

Print **Relay_1: Status**

Print **Relay_2: Status**

Update Relay_1 & Realay_2 Status on IoT Firebase Database

7. else

Stop Program

PROGRAM

PROGRAM	COMMENT
import lcddriver	Import LCD Library
from time import *	Import Delay Library
lcd = lcddriver.lcd()	Initialize LCD
import RPi.GPIO as GPIO	import GPIO library
import pyrebase	Import Firebase Library
from time import sleep	Import Delay Library
import time	Import Time Library

config = {
 	
"apiKey": "wgsNjBl7qvzVY0KiezCBplSlRxLI9OiPwEschMfh",
"authDomain": "rasptest-2fdb7.firebaseapp.com",
"databaseURL": "https://rasptest-2fdb7-default-rtdb.firebaseio.com/",
"storageBucket": "rasptest123"
}
	configure firebase database
	Variables

firebase = pyrebase.initialize_app(config)	Initialize Firebase
LED_1 = 18	LED_1 connect with GPIO 18
LED_2 = 23	LED_2 connect with GPIO 23
GPIO.setmode(GPIO.BCM)	Define as Physical numbering
GPIO.setwarnings(False)	Warning Off for GPIO pins
GPIO.setup(LED_1,GPIO.OUT)	Configure LED_1 as Output
GPIO.setup(LED_2,GPIO.OUT)	Configure LED_2 as Output

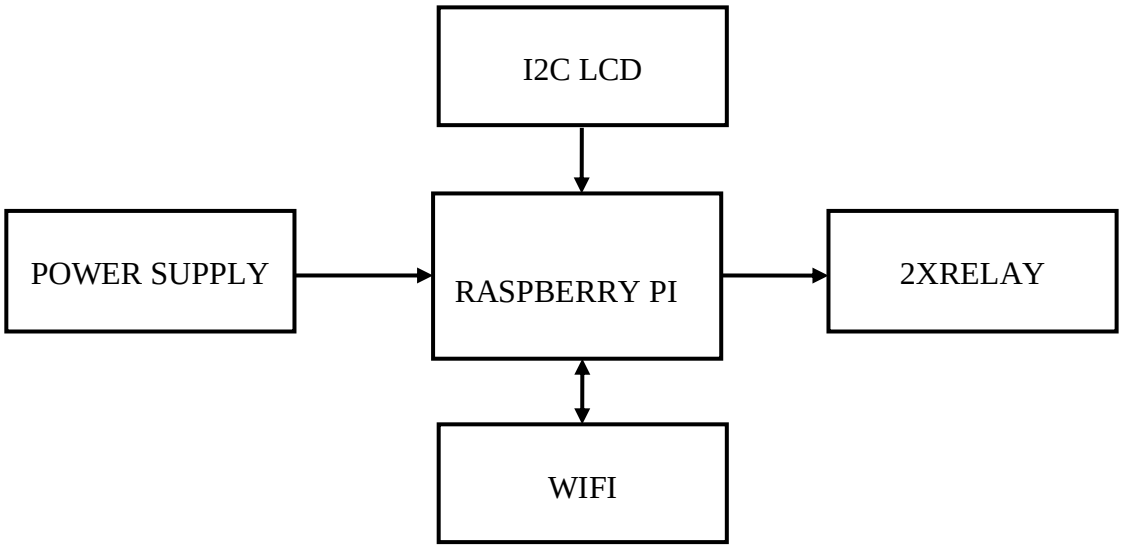
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" IoT Based ", 1)	Print **IoT Based** on LCD 1st Line
lcd.lcd_display_string("Home Automation" , 2)	Print **Home Automation** on LCD 2nd Line
time.sleep(2)	Delay 2 Second
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" System", 1)	Print **System** on LCD 1st Line
lcd.lcd_display_string(" *******" , 2)	Print ********** on LCD 2nd Line
time.sleep(2)	Delay 2 Second
try:	Test a block of code for errors
while True:	Infinite Loop Start
database = firebase.database()	Read Firebase Database
ProjectBucket = database.child("IOTLAB")	Define Firebase Data Storage Bucket
LEDDATA1 = ProjectBucket.child("LED1").get().val()	Read LED_1 Data
ProjectBucket = database.child("IOTLAB")	Define Firebase Data Storage Bucket
LEDDATA2 = ProjectBucket.child("LED2").get().val()	Read LED_2 Data
lcd.lcd_clear()	Clear LCD
if str(LEDDATA1) == "1":	Check LED_1 ON Loop
print("LED_1 now is ON.")	Print **LED_1 now is ON** on Console
GPIO.output(LED_1, GPIO.HIGH)	LED_1 ON
ProjectBucket.child("IOTLAB").child("L1").set("ON")	Update LED1 Status on IoT as **ON**
lcd.lcd_display_string("LED_1 : ON", 1)	Print **LED-1: ON** on LCD 1st Line
else:	Check LED_1 OFF Loop
print("LED_1 now is OFF.")	Print **LED_1 now is OFF** on Console
GPIO.output(LED_1, GPIO.LOW)	LED_1 OFF
ProjectBucket.child("IOTLAB").child("L1").set("OFF")	Update LED1 Status on IoT as **OFF**
lcd.lcd_display_string("LED_1 : OFF", 1)	Print **LED-1: OFF** on LCD 2nd

	Line
if str(LEDDATA2) == "1":	Check LED_2 ON Loop
print("LED_2 now is ON.")	Print **LED_2 now is ON** on Console
GPIO.output(LED_2, GPIO.HIGH)	LED_2 ON
ProjectBucket.child("IOTLAB").child("L2").set("ON	Update LED2 Status on IoT as **ON**
lcd.lcd_display_string("LED_2 : ON", 2)	Print **LED_2: ON** on LCD 1st Line
else:	Check LED_2 OFF Loop
print("LED_2 now is OFF.")	Print **LED_2 now is OFF** on Console
GPIO.output(LED_2, GPIO.LOW)	LED_2 OFF
ProjectBucket.child("IOTLAB").child("L2").set("OFF")	Update LED2 Status on IoT as **OFF**
lcd.lcd_display_string("LED_2 : OFF", 2)	Print **LED_2: OFF** on LCD 2nd Line
time.sleep(2)	Delay 2 Second
except KeyboardInterrupt:	Program Is Interrupted By The User Keyboard
GPIO.cleanup()	Reset GPIO Pins

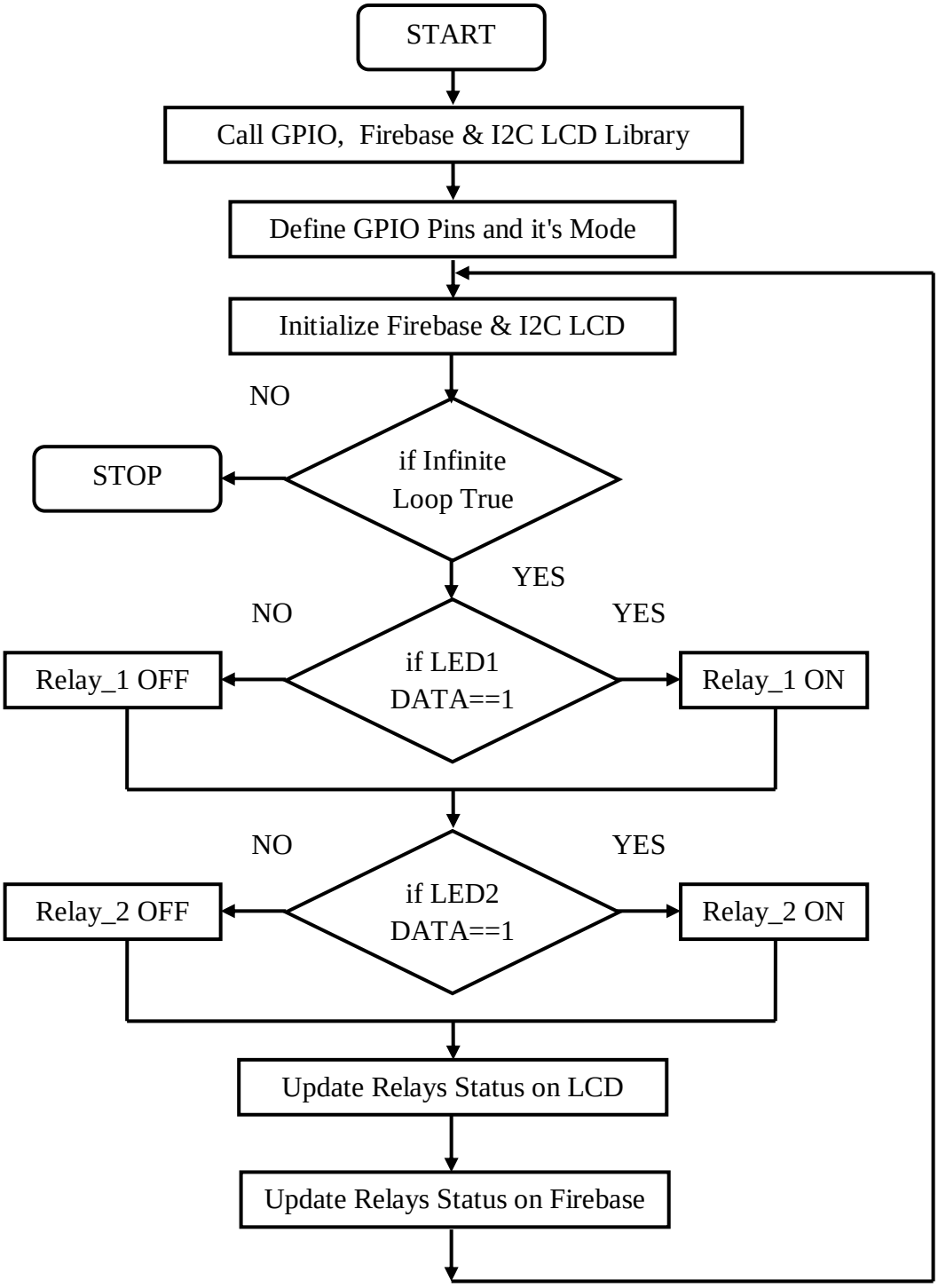
CONNECTION TABLE

S.NO	CONNECTION		PIN MODE
	COMPONENT	RASPBERRY PI	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	Relay_1 (R1)	GPIO 18	OUTPUT
4	Relay_2 (R2)	GPIO 23	OUTPUT
5	LCD (SDA)	GPIO 2	Serial Data
6	LCD (SCL)	GPIO 3	Serial Clock

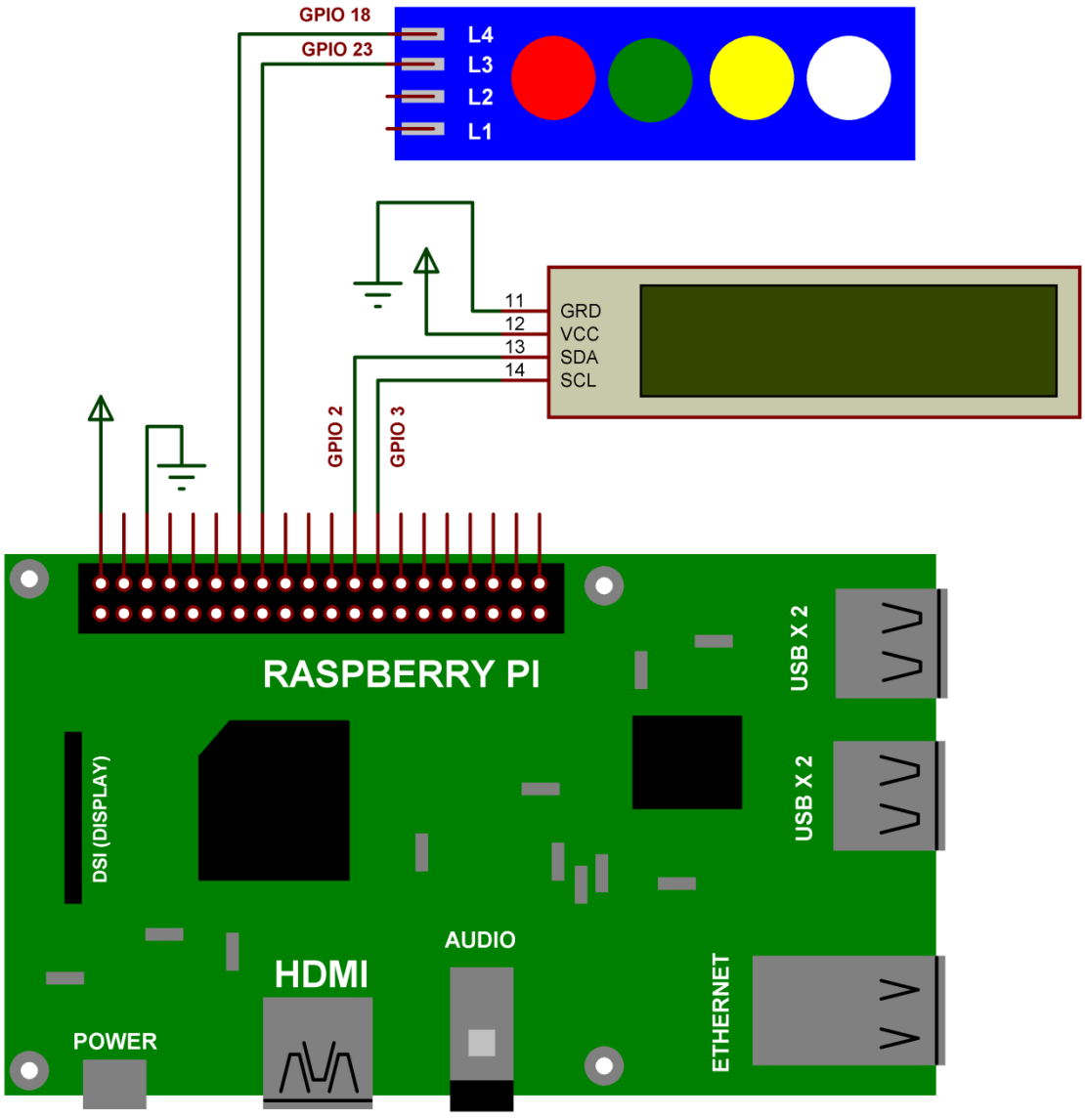
BLOCK DIAGRAM



FLOW CHART


START
Call GPIO, Firebase & I2C LCD Library
Define GPIO Pins and it's Mode
Initialize Firebase & I2C LCD
NO
STOP
if Infinite Loop True
YES
NO
YES
Relay_1 OFF
if LED1 DATA==1
Relay_1 ON
NO
YES
Relay_2 OFF
if LED2 DATA==1
Relay_2 ON
Update Relays Status on LCD
Update Relays Status on Firebase

CIRCUIT DIAGRAM



TERMINAL COMMAND LINE

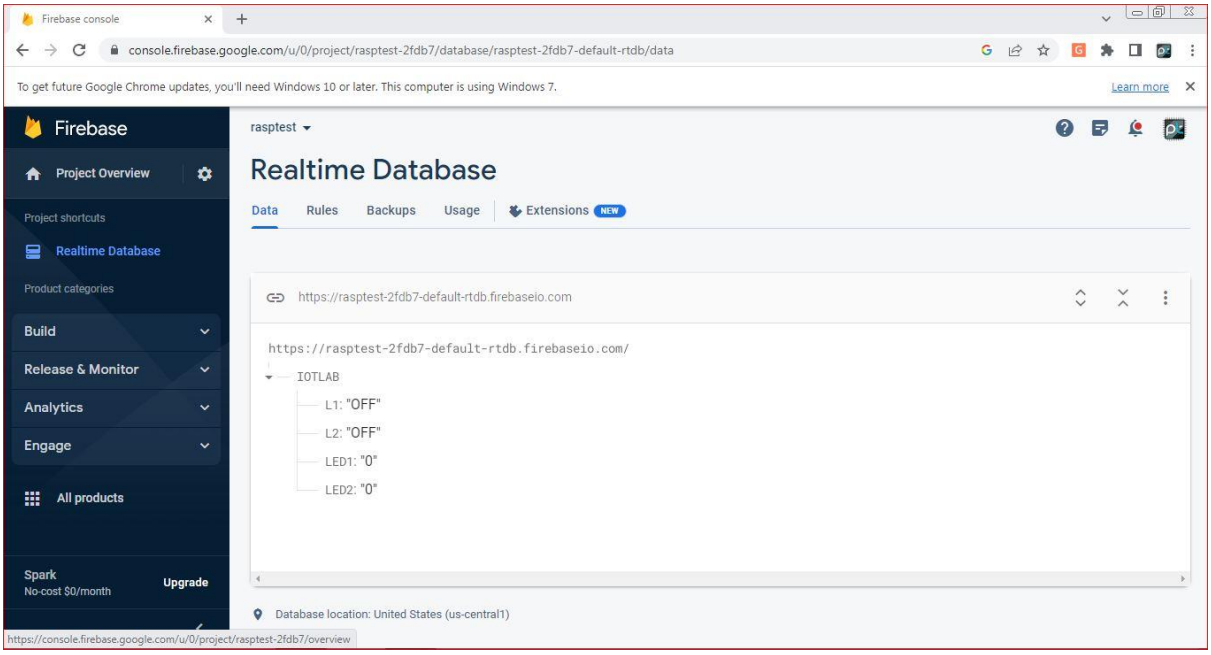
1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo raspi-config
4. Select Interfacing Options > I2C.

5. Select Yes when prompted to enable the I2C interface.
6. Select Yes when prompted to automatically load the I2C kernel module.
7. Select Finish.
8. i2cdetect -y 1
9. sudo raspi-config
10. Select Interfacing Options >1 WIRE.
11. Select Yes when prompted to enable the 1 WIRE interface.
12. Select Yes when prompted to automatically load the 1 WIRE kernel module.
13. Select Finish.
14. sudo apt-get update.
15. sudo apt-get install python-dev.
16. sudo python get-pip OR sudo apt-get install python-pip (new Raspian versions)
17. sudo pip install pyrebase.

PROCEDURE:

1. Start **Thonny** by clicking on the **Raspberry Pi icon** followed by **Programming > Thonny Python IDE.**
2. Write your program in the top pane, click **File > Save** as... to save it,
3. Make hardware connections
4. Click **Run** > Run current script to execute the program. Output will appear in the bottom interpreter pane.

RESULT SCREEN SHOT



Result