EXP.NO:1 DATE:	WRITE 8051 ASSEMBLY LANGUAGE EXPERIMENTS USING SIMULATOR.

AIM:

To write and execute Assembly Language Program for Addition, Subtraction, Multiplication and Division of two 8-Bit numbers using EDSIM51 simulator.

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	EDSIM51 Software	1

ALGORITHM:

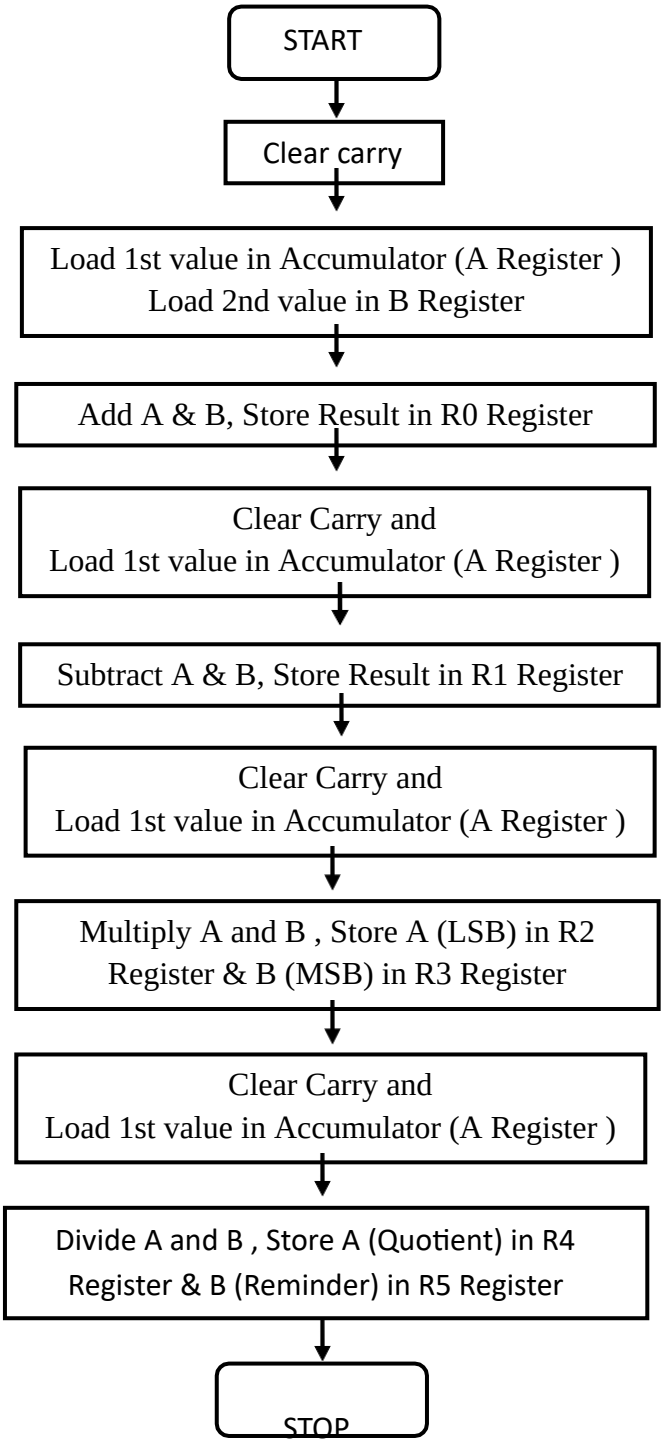
1. Start the program.
2. Clear carry
3. Load 1st value in Accumulator (A Register)
4. Load 2nd value in B register
5. Add Accumulator and B register value and result store in the Accumulator
6. Store the resultant value in R0 register
7. Clear carry
8. Load 1st value in Accumulator (A Register)
9. Subtract Accumulator and B register value and result store in the Accumulator
10. Store the resultant value in R1 register
11. Clear carry
12. Load 1st value in Accumulator (A Register)
13. Multiply Accumulator and B register value and LSB value Stored in Register ,MSB value Stored in B Register

14. 14 Store the resultant value LSB in R2 Register and MSB in R3Register
15. Clear carry
16. Load 1st value in Accumulator (A Register)
17. Divide Accumulator and B register value and Quotient value Stored in A Register , Reminder value Stored in B Register
18. Store the resultant value Quotient in R4 Register and Reminder in R5 Register
19. Stop the program

PROGRAM:

PROGRAM	COMMENT
clr c	Clear Carry
mov a, #33h	Load 1st Data in Accumulator
mov b, #22h	Load 2nd Data in B Register
add a,b	Add A & B
mov r0, a	Store Data in R0 Register
clr c	Clear Carry
mov a, #33h	Load 1st Data in Accumulator
subb a,b	Subtract A & B
mov r1, a	Store Data in R1 Register
clr c	Clear Carry
mov a, #33h	Load 1st Data in Accumulator
mul ab	Multiply A & B
mov r2, a	Store LSB Data in R2 Register
mov r3, b	Store MSB Data in R3 Register
clr c	Clear Carry
mov a, #33h	Load 1st Data in Accumulator
div ab	Divide A & B
mov r4, a	Store Quotient Data in R4 Register
mov r5, b	Store Reminder Data in R5 Register
end	Stop Program

FLOW CHART:


START
Clear carry
Load 1st value in Accumulator (A Register)
Load 2nd value in B Register
Add A & B, Store Result in R0 Register
Clear Carry and
Load 1st value in Accumulator (A Register)
Subtract A & B, Store Result in R1 Register
Clear Carry and
Load 1st value in Accumulator (A Register)
Multiply A and B , Store A (LSB) in R2
Register & B (MSB) in R3 Register
Clear Carry and
Load 1st value in Accumulator (A Register)
Divide A and B , Store A (Quotient) in R4
Register & B (Reminder) in R5 Register
STOP

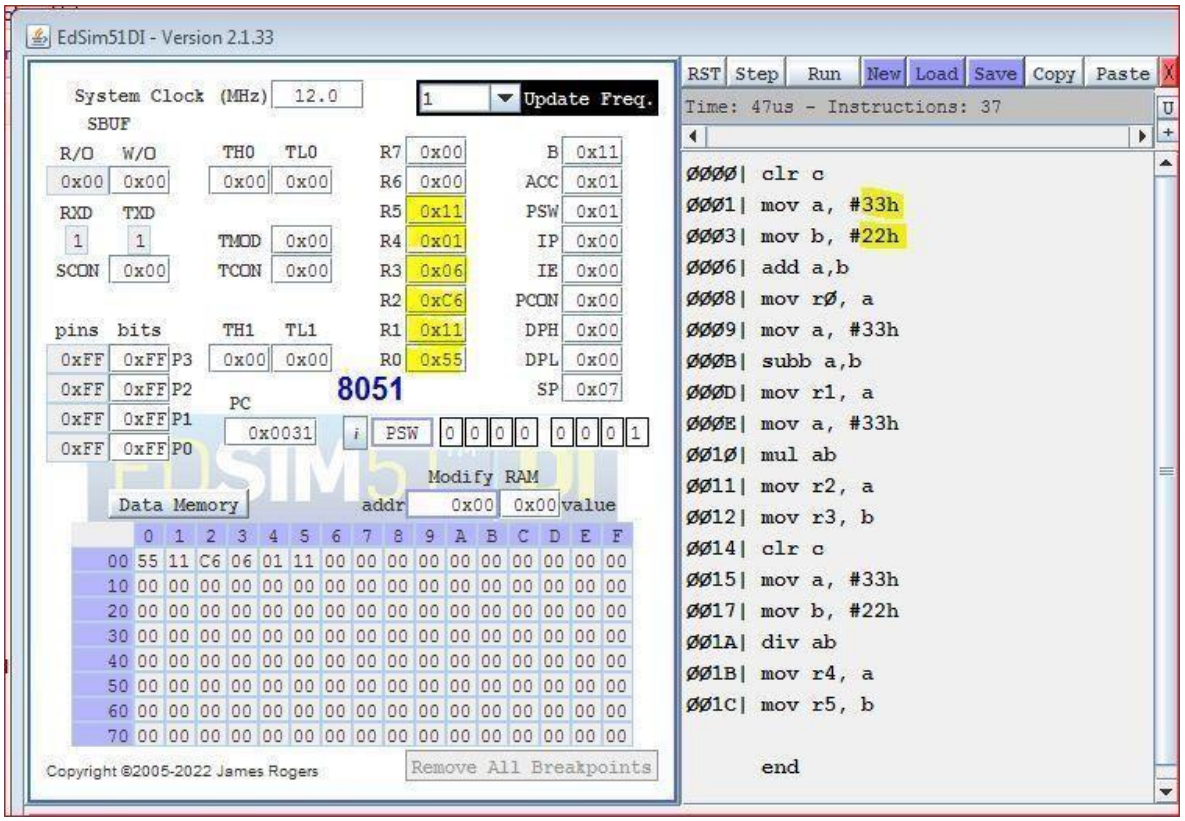
PROCEDURE:

1. Open EDSIM51 Software
2. Type the assembly language in Programming Section
3. Click Assm Button to compile the program (If any error present click RST Button and debug the program)
4. Click Run Button to execute the program
5. Verify the result in Registers and Internal RAM section
6. Click Pause Button to Stop the program

EXAMPLE:	**RESULT:**
Addition : **A = 33h = 0011 0011_2** **B = 22h = 0010 0010_2**	
R0= 55h = 0101 0101_2	
Subtraction : **A = 33h = 0011 0011_2** **B = 22h = 0010 0010_2**	
R1= 11h = 0001 0001_2	
Multiplication : **A = 33h = 0011 0011_2** **B = 22h = 0010 0010_2**	
RESULT = 06 C6h = 0000 0110 1100 0110_2 **LSB = R2 = C6 h = 1100 0110_2** **MSB = R3 = 06 h = 0000 0110_2**	

Division : **A = 33h = 0011 0011_2** **B = 22h = 0010 0010_2**	
Quotient = R4 = 01h = 0000 0001_2 **Reminder = R5 = 11 h = 0001 0001_2**	

RESULT SCREENSHOT:



EXP.NO:2 DATE:	TEST DATA TRANSFER BETWEEN REGISTERS AND MEMORY OF TWO 8-BIT NUMBERS IN ASSEMBLY LANGUAGE

AIM:

To write and execute Assembly Language Program for Circular Logic Rotate Right , Circular Logic Rotate Left and Swap Operations of 8-Bit number using EDSIM51 simulator.

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	EDSIM51 Software	1

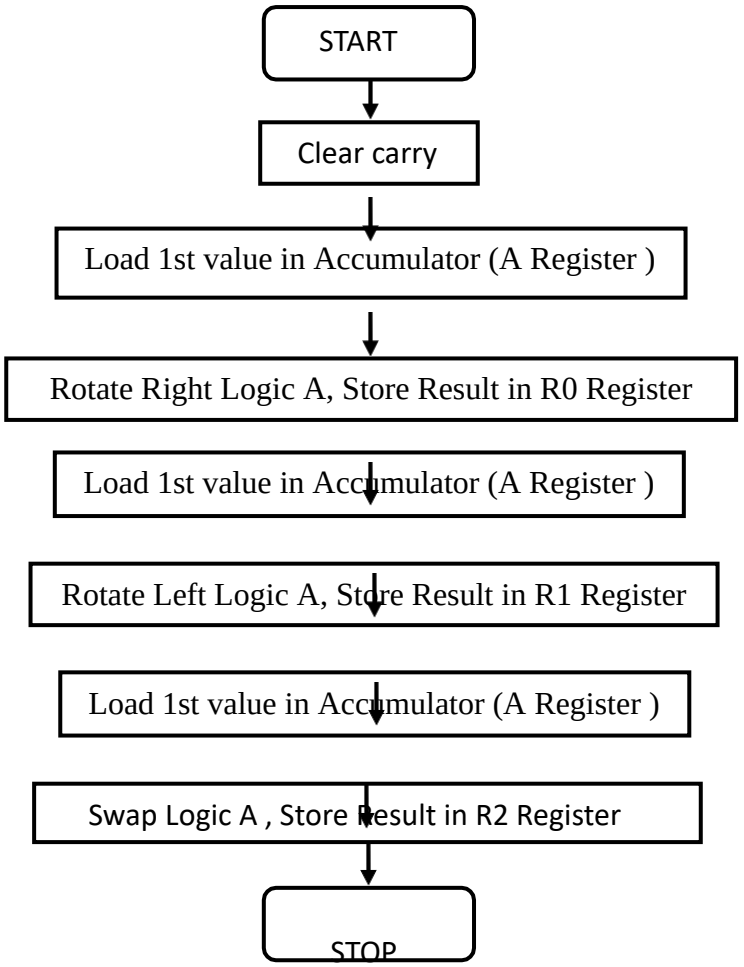
ALGORITHM:

1. Start the program.
2. Clear carry
3. Load 1st value in Accumulator (A Register)
4. Rotate Right Logic in Accumulator and result store in the Accumulator
6. Store the resultant value in R0 register
7. Load 1st value in Accumulator (A Register)
8. Rotate Left Logic in Accumulator and result store in the Accumulator
9. Store the resultant value in R1 register
10. Load 1st value in Accumulator (A Register)
11. Swap Logic in Accumulator and result store in the Accumulator
12. Store the resultant value in R2 Register
14. Stop the program.

PROGRAM:

PROGRAM	COMMENT
clr c	Clear Carry
mov a, #33h	Load 1st Data in Accumulator
rr a	Rotate Right Logic A
mov r0, a	Store Data in R0 Register
mov a, #33h	Load 1st Data in Accumulator
rl a	Rotate Right Logic A
mov r1, a	Store Data in R1 Register
mov a, #33h	Load 1st Data in Accumulator
swap a	Swap Logic A
mov r2, a	Store Data in R2 Register
end	Stop Program

FLOW CHART:

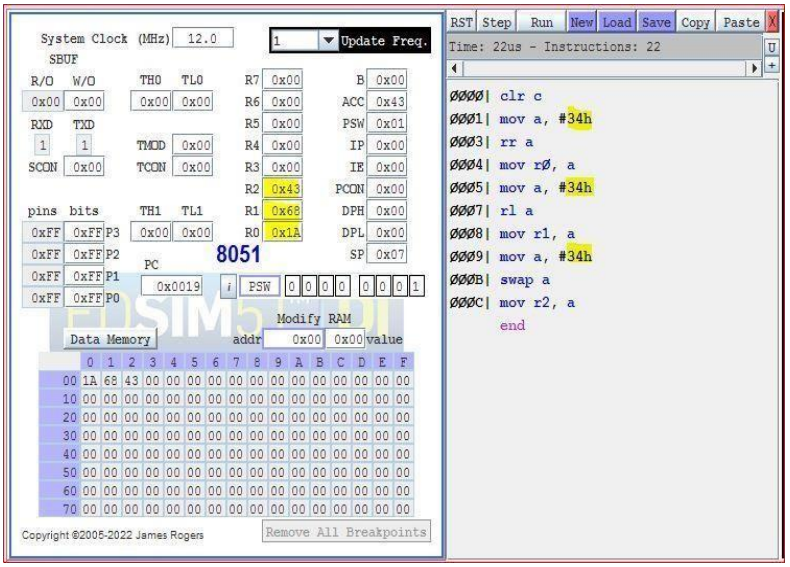


PROCEDURE:

1. Open EDSIM51 Software
2. Type the assembly language in Programming Section
3. Click Assm Button to compile the program (If any error present click RST Button and debug the program)
4. Click Run Button to execute the program
5. Verify the result in Registers and Internal RAM section
6. Click Pause Button to Stop the program

EXAMPLE:	**RESULT:**
ROTATE RIGHT: **A = 34h = 0011 0110$_2$**	
R0= 1Ah = 0001 1010$_2$	
ROTATE LEFT: **A = 34h = 0011 0110$_2$**	
R1= 68h = 0110 1000$_2$	
SWAP: **A = 34h = 0011 0110$_2$**	
R2 = 43h = 0100 0011$_2$	

RESULT SCREENSHOT:



EXP.NO:3 DATE:	PERFORM ALU LOGICAL OPERATION OF TWO 8-BIT NUMBERS IN ASSEMBLY LANGUAGE

AIM:

To write and execute Assembly Language Program for Addition, Subtraction, Multiplication and Division of two 8-Bit numbers using EDSIM51 simulator.

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	EDSIM51 Software	1

ALGORITHM:

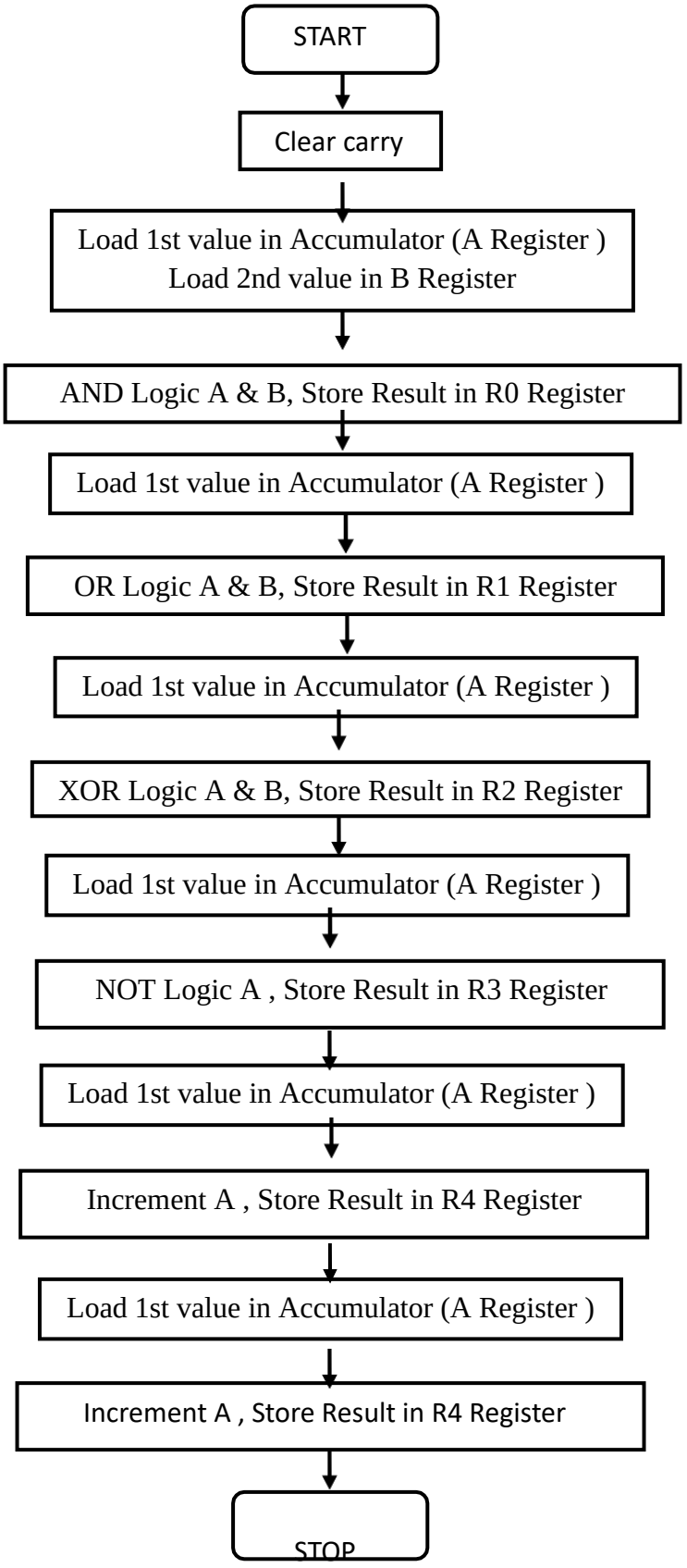
1. Start the program.
2. Clear carry
3. Load 1st value in Accumulator (A Register)
4. Load 2nd value in B register
5. AND Logic between Accumulator and B register value and result store in the Accumulator
6. Store the resultant value in R0 register
7. Load 1st value in Accumulator (A Register)
8. OR Logic between Accumulator and B register value and result store in the Accumulator
9. Store the resultant value in R1 register
10. Load 1st value in Accumulator (A Register)
11. XOR Logic between Accumulator and B Register value and result store in the Accumulator

12. Store the resultant value in R2 Register
13. Load 1st value in Accumulator (A Register)
14. NOT Logic between Accumulator value and result store in the Accumulator
15. Store the resultant value in R3 Register
16. Load 1st value in Accumulator (A Register)
17. Perform Increment Operation in Accumulator value and result store in the Accumulator
18. Store the resultant value in R4 register
19. Load 1st value in Accumulator (A Register)
20. Perform Decrement Operation in Accumulator value and result store in the Accumulator
21. Store the resultant value in R5 register
22. Stop the program.

PROCEDURE:

1. Open EDSIM51 Software
2. Type the assembly language in Programming Section
3. Click Assm Button to compile the program (If any error present click RST Button and debug the program)
4. Click Run Button to execute the program
5. Verify the result in Registers and Internal RAM section
6. Click Pause Button to Stop the program

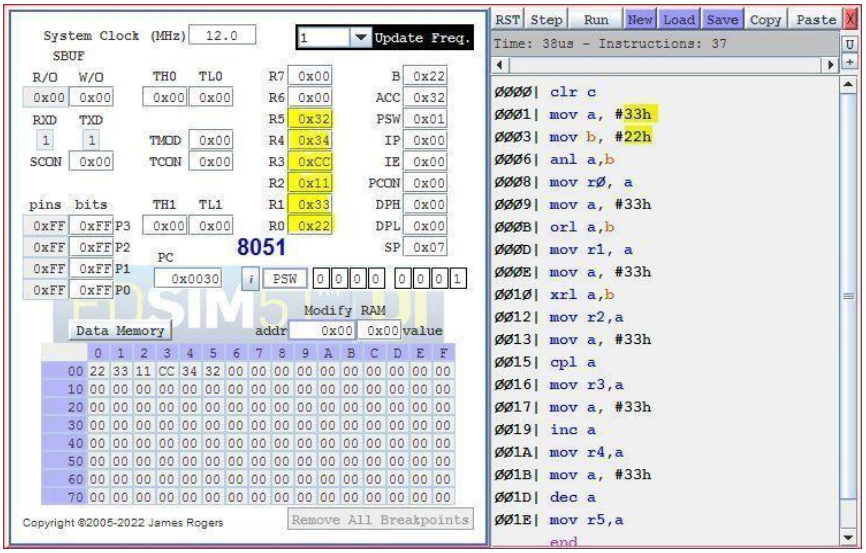
FLOW CHART


START
Clear carry
Load 1st value in Accumulator (A Register)
Load 2nd value in B Register
AND Logic A & B, Store Result in R0 Register
Load 1st value in Accumulator (A Register)
OR Logic A & B, Store Result in R1 Register
Load 1st value in Accumulator (A Register)
XOR Logic A & B, Store Result in R2 Register
Load 1st value in Accumulator (A Register)
NOT Logic A , Store Result in R3 Register
Load 1st value in Accumulator (A Register)
Increment A , Store Result in R4 Register
Load 1st value in Accumulator (A Register)
Increment A , Store Result in R4 Register
STOP

EXAMPLE:	RESULT:
AND: **A = 33h = 0011 00112** **B = 22h = 0010 00102**	
R0= 22h = 0010 00102	
OR: **A = 33h = 0011 00112** **B = 22h = 0010 00102**	
R1= 33h = 0011 00112	
XOR : **A = 33h = 0011 00112** **B = 22h = 0010 00102**	
R2 = 11h = 0001 00012	
NOT: **A = 33h = 0011 00112**	

R3 = CCh = 1100 11002	
INCREMENT: **A = 33h = 0011 00112**	
R4 = 34h = 0011 01002	
DECREMENT: **A = 33h = 0011 00112**	
R5 = 32h = 0011 00102	

RESULT SCREEN SHOT:



EXP.NO:4 DATE:	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

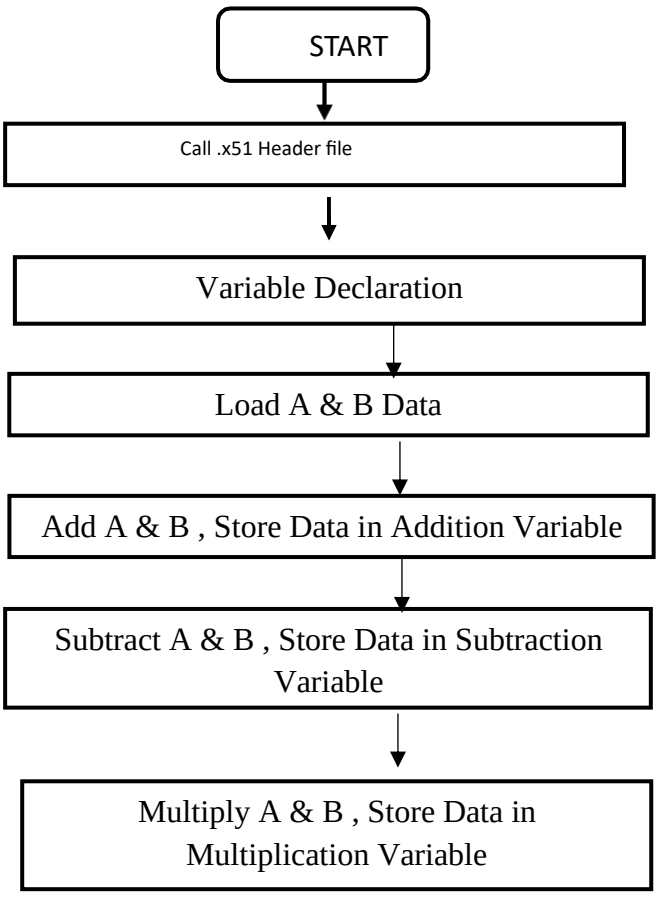
ALGORITHM:

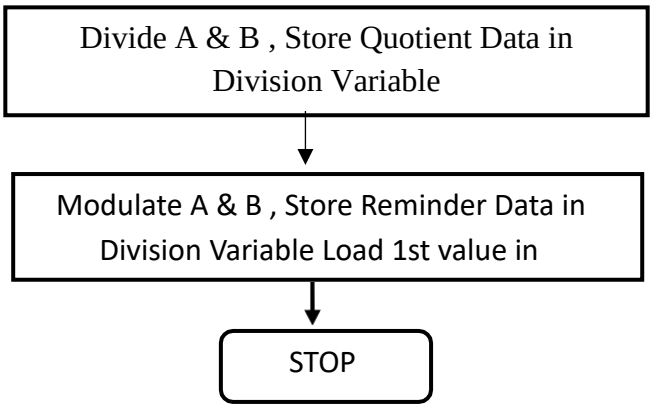
1. Start the program.
2. Call x51 header file
3. Declare Variables
3. Load 1st value in A
4. Load 2nd value in B
5. Add A and B values and result store in the Addition variable
6. Subtract A and B values and result store in the Subtraction variable
7. Multiply A and B values and result store in the Multiplication 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





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**.

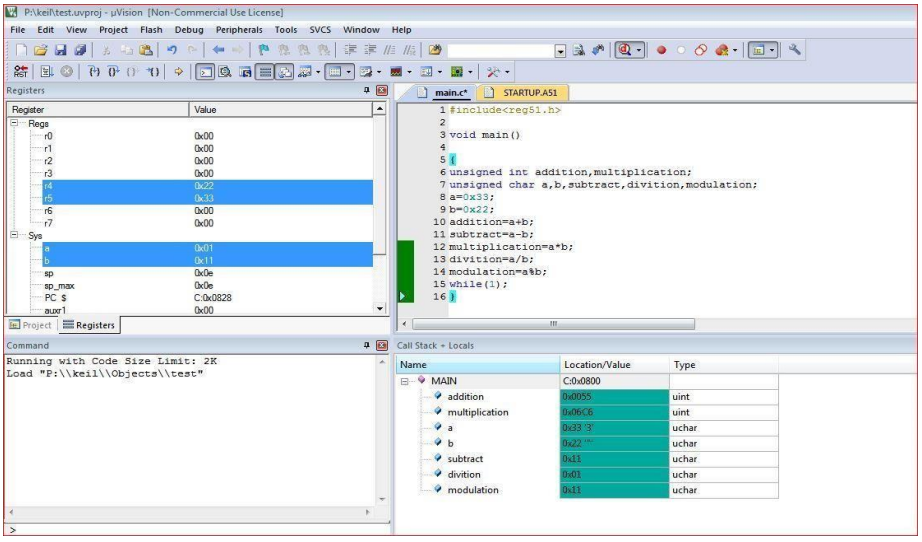
8. Verify Source File is added or not by Expand the **'Source Group 1'** folder in Project Section.
9. Type the Program and Save it.
10. Click **Project >> Rebuilt all target files.** (If error presents debug the program & Click again **Project >> Rebuilt all target files** . Repeat this process until 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.

EXAMPLE:	RESULT:
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:



EXP.NO:5 DATE:	INTRODUCTION TO ARDUINO PLATFORM AND PROGRAMMING

AIM:

To write introduction on Arduino platform and programming

ARDUINO PLATFORM:

The Arduino platform is based on a microcontroller board and a development environment for writing software for the board. The microcontroller is the brain of the Arduino, and it is responsible for interpreting and executing the code written by the user. The development environment, on the other hand, is a software tool that allows users to write, compile, and upload code to the Arduino board.

The programming language used for Arduino is based on Wiring, a similar language to C/C++. It is easy to learn and use, making it accessible to beginners and experienced programmers alike. The code written for Arduino is called a sketch, and it consists of functions that define what the board should do.

Arduino boards come in various shapes and sizes, each with different features and capabilities. Some boards are designed for simple projects with few inputs and outputs, while others are more powerful and can handle more complex tasks. It is important to choose the right board for your project based on its requirements and your budget.

In addition to the hardware, Arduino also has a vast community of users who share their projects, code, and knowledge online. This community is a valuable resource for learning and troubleshooting, as well as for finding inspiration for new projects.

Additional points about the Arduino platform:

1. Open-source: Arduino is an open-source platform, which means that the hardware and software designs are freely available for anyone to use, modify, and distribute. This has led to a large and active community of developers and enthusiasts who contribute to the platform's growth and innovation.

2. Extensibility: Arduino boards can be expanded and customized through the use of shields, which are add-on boards that provide additional functionality such as wireless communication, motor control, and sensor inputs. This allows users to tailor their Arduino projects to specific requirements without having to design and build custom hardware from scratch.

3. Interactivity: One of the key features of Arduino is its ability to interact with the physical world through various inputs and outputs. This makes it well-suited for creating interactive art installations, robotics projects, home automation systems, and more.

4. Education: Arduino is widely used in educational settings to teach students about electronics, programming, and physical computing. Its accessibility and ease of use make it an ideal platform for introducing beginners to the world of technology and engineering.

5. Integration: Arduino can be integrated with other software and hardware platforms, such as Raspberry Pi, Processing, and various IoT (Internet of Things) platforms. This allows for even greater flexibility and interoperability in creating complex and interconnected systems.

6. Industry adoption: While Arduino is often associated with hobbyist and educational projects, it is also used in professional and industrial settings for prototyping, testing, and even as a component in commercial products.

Overall, Arduino's combination of simplicity, flexibility, and community support makes it a valuable tool for anyone interested in exploring the intersection of technology and creativity. Whether you're a tinkerer, a student, or a professional engineer, Arduino offers a rich ecosystem for bringing your ideas to life.

EXP.NO:6(a) DATE:	**EXPLORE DIFFERENT COMMUNICATION METHODS WITH IOT DEVICES (ZIGBEE)**

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

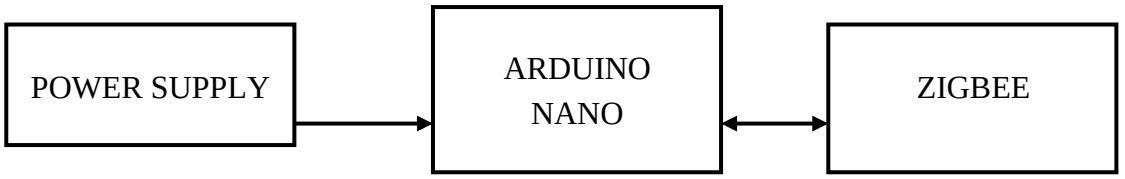
PROGRAM

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

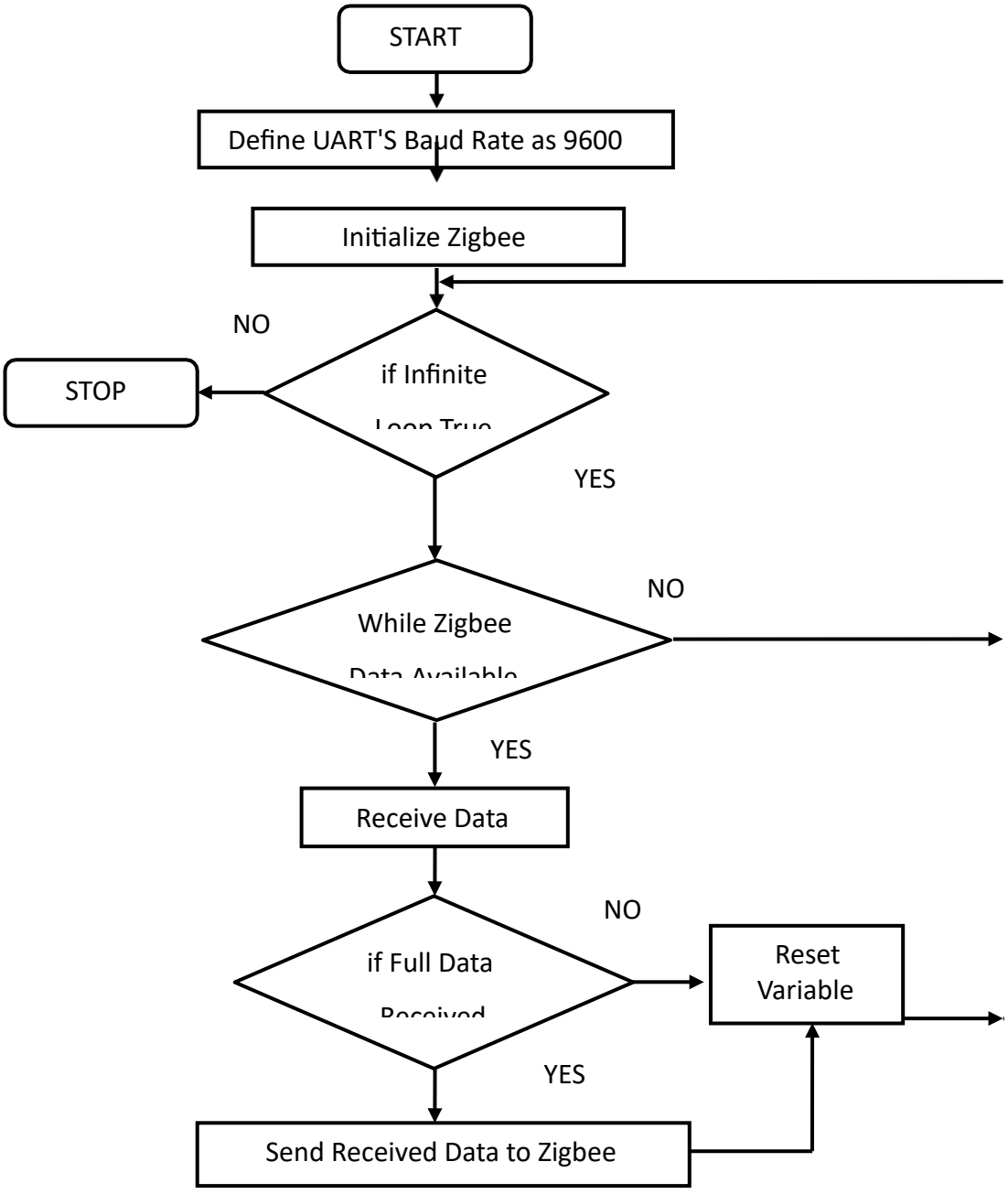
CONNECTION TABLE

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

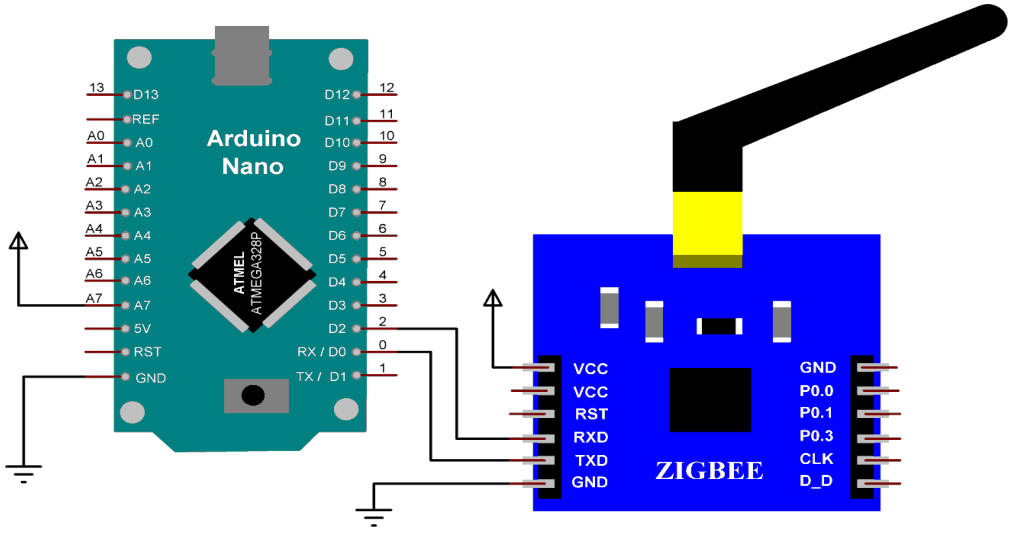
BLOCK DIAGRAM



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.

EXP.NO:6(b) DATE:	EXPLORE DIFFERENT COMMUNICATION METHODS WITH IOT DEVICES (GSM)

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

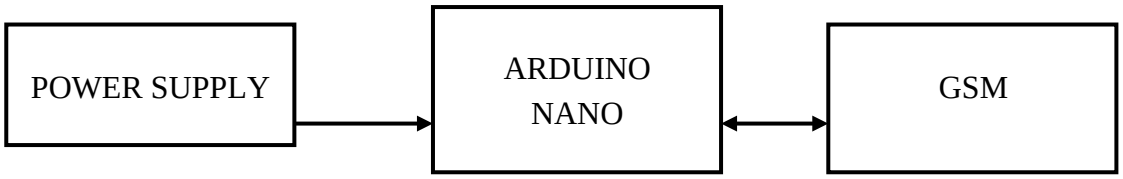
PROGRAM

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

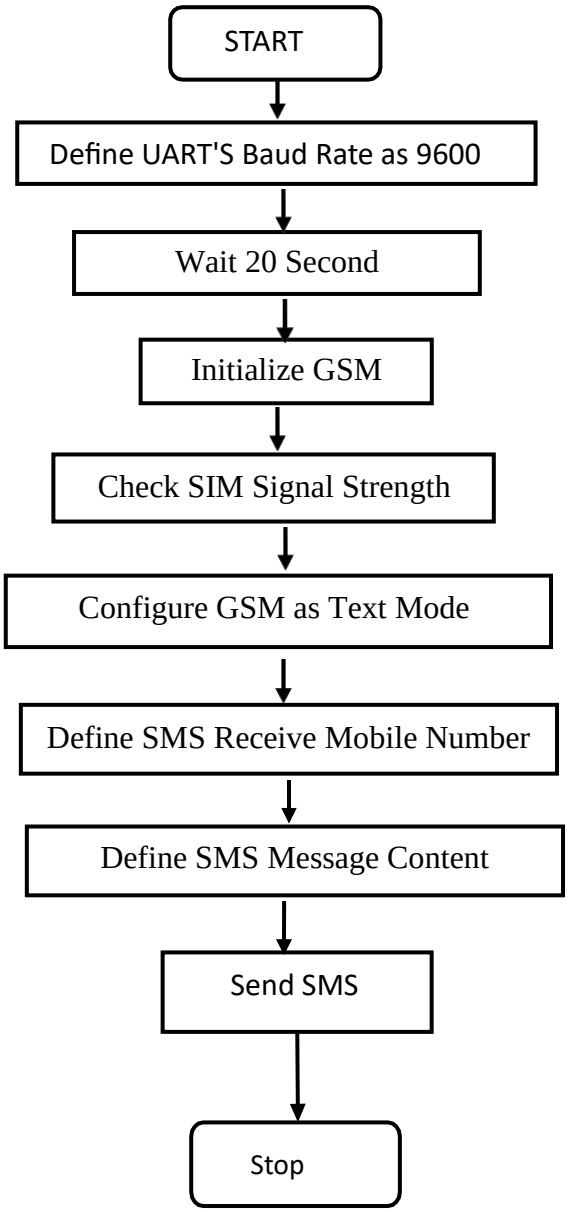
CONNECTION TABLE

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

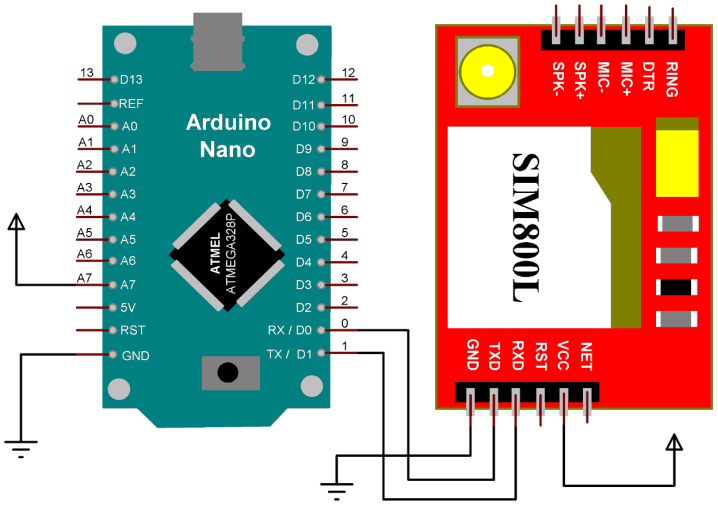
BLOCK DIAGRAM



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.

EXP.NO:6(c) DATE:	EXPLORE DIFFERENT COMMUNICATION METHODS WITH IOT DEVICES (BLUETOOTH)

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

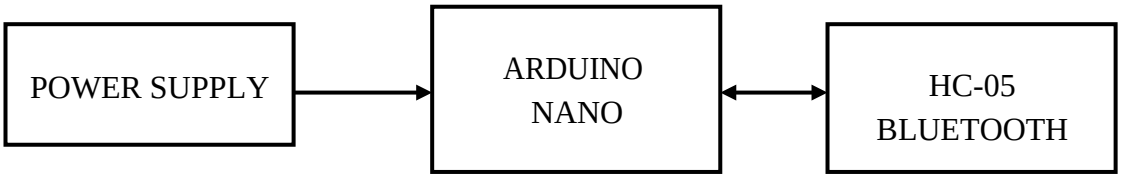
PROGRAM

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

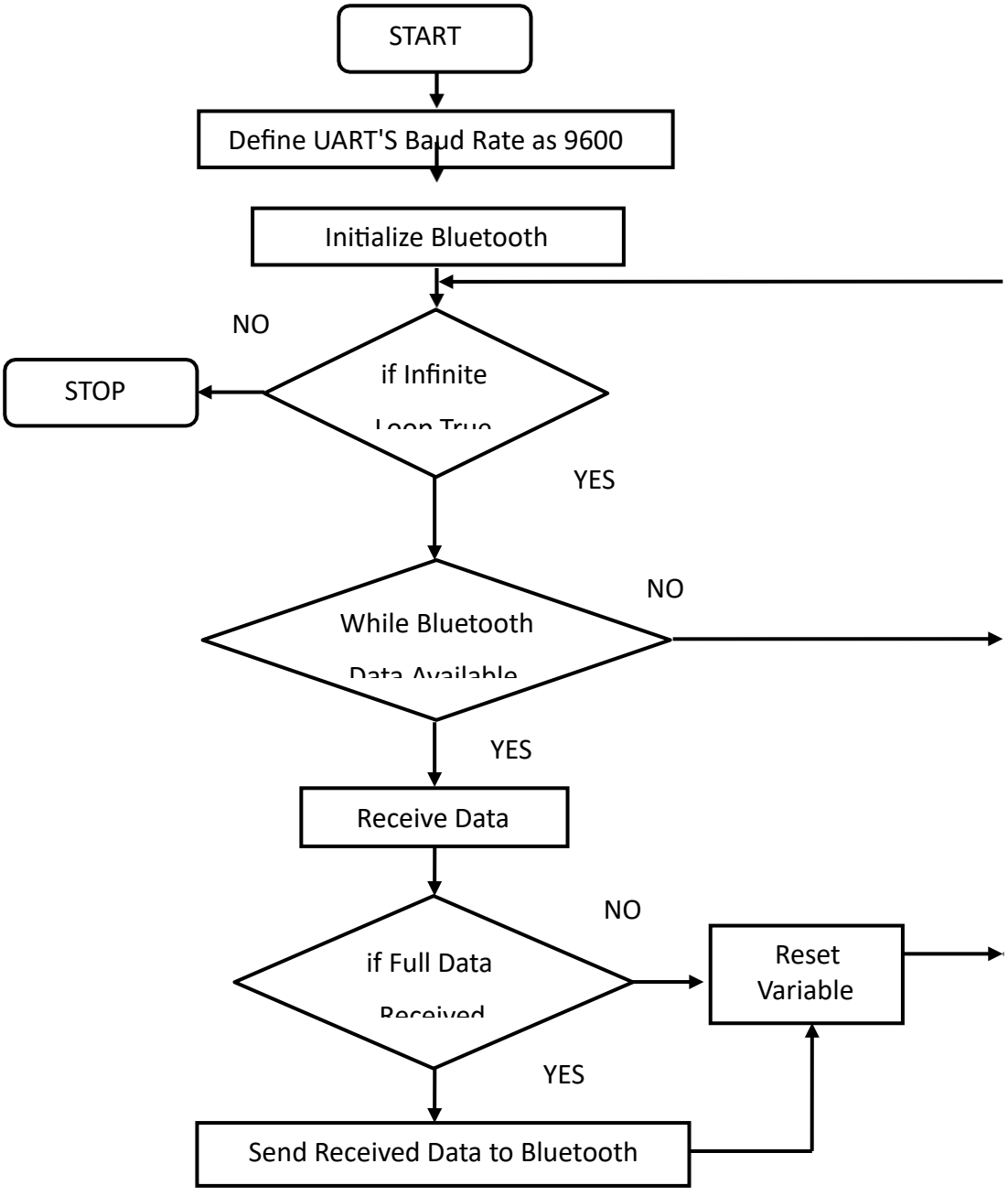
CONNECTION TABLE

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

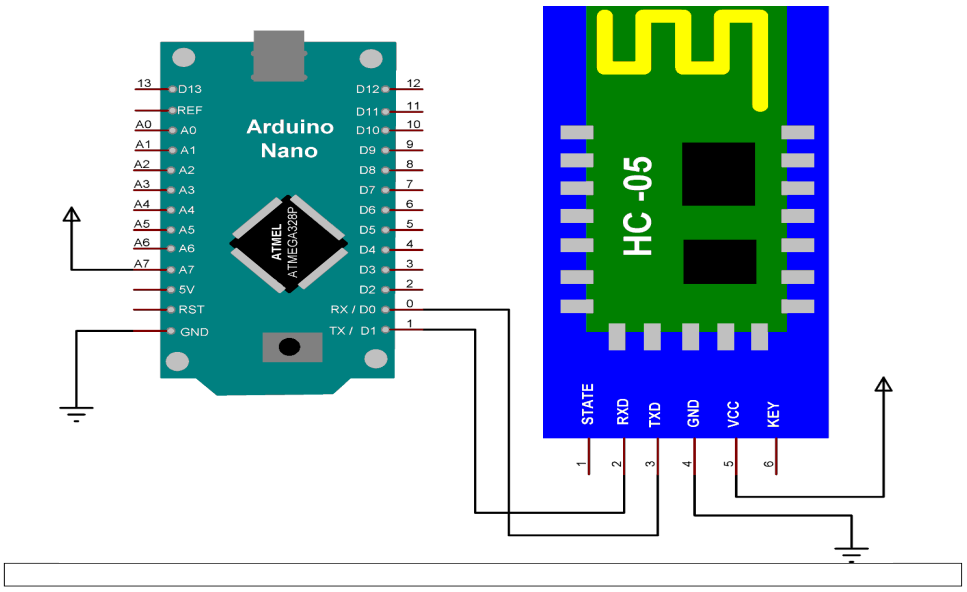
BLOCK DIAGRAM



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.

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

14.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

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

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

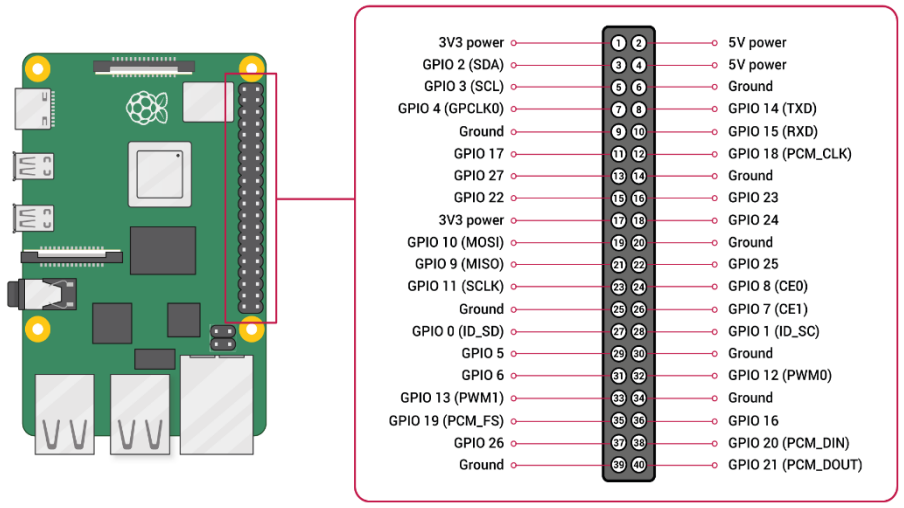
EXP.NO:7 DATE:	INTRODUCTION TO RASPBERRY PI PLATFORM AND PYTHON PROGRAMMING

AIM:

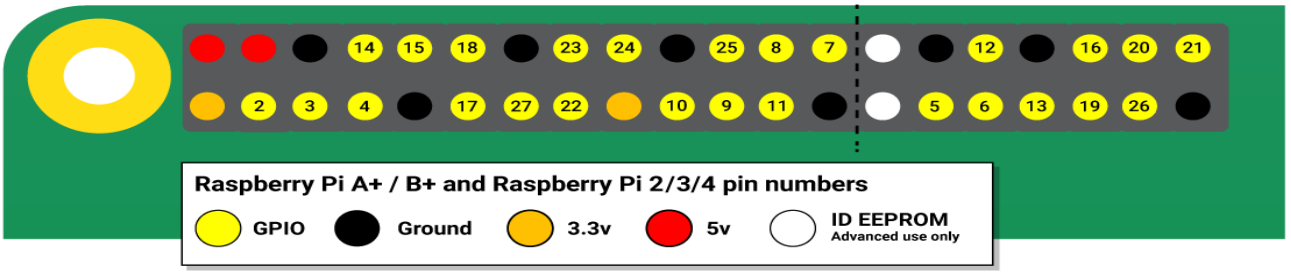
To introduce raspberry pi platform and python programming.

GPIO and the 40-pin Header

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Raspberry Pi Zero, Raspberry Pi Zero W and Raspberry Pi Zero 2 W). Prior to the Raspberry Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header. The GPIO header on all boards (including the Raspberry Pi 400) have a 0.1" (2.54mm) pin pitch.



Any of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purposes.



Voltages

Two 5V pins and two 3.3V pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3.3V pins, meaning outputs are set to 3.3V and inputs are 3.3V-tolerant.

Outputs

A GPIO pin designated as an output pin can be set to high (3.3V) or low (0V).

Inputs

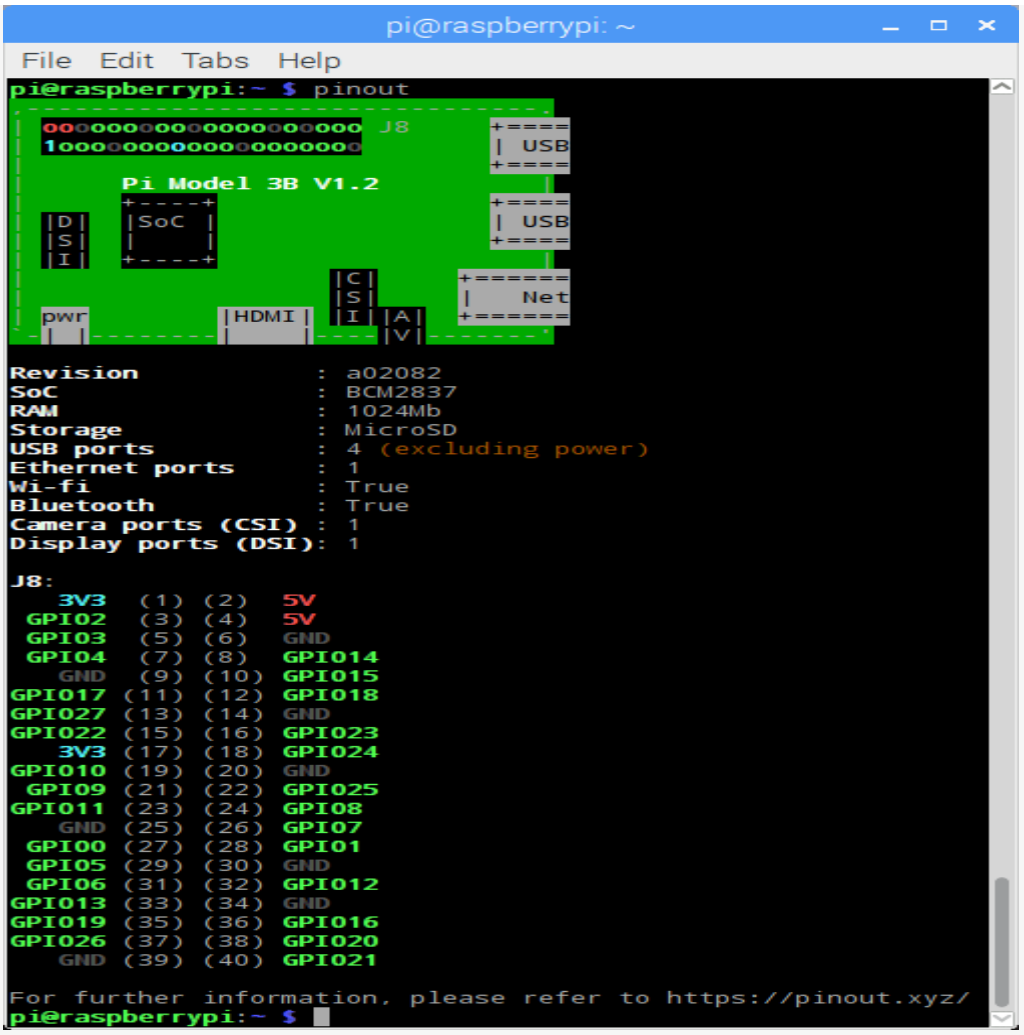
A GPIO pin designated as an input pin can be read as high (3.3V) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

As well as simple input and output devices, the GPIO pins can be used with a variety of alternative functions, some are available on all pins, others on specific pins.

- PWM (pulse-width modulation)
    - Software PWM available on all pins
    - Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
- SPI
    - SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
    - SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
- I2C
    - Data: (GPIO2); Clock (GPIO3)
    - EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)
- Serial
    - TX (GPIO14); RX (GPIO15)

GPIO pinout

A handy reference can be accessed on the Raspberry Pi by opening a terminal window and running the command pinout. This tool is provided by the GPIO Zero Python library, which is installed by default in Raspberry Pi OS.



In order to use the GPIO ports your user must be a member of the gpio group. The pi user is a member by default, other users need to be added manually.

```
sudo usermod -a -G gpio <username>
```

GPIO in Python

Using the GPIO Zero library makes it easy to get started with controlling GPIO devices with Python. The library is comprehensively documented at gpiozero.readthedocs.io.

LED

To control an LED connected to GPIO17, you can use this code:

```
from gpiozero import LED
from time import sleep

led = LED(17)

while True:
    led.on()
    sleep(1)
    led.off()
```

```
    sleep(1)
```

Run this in an IDE like Thonny, and the LED will blink on and off repeatedly.

LED methods include `on()`, `off()`, `toggle()`, and `blink()`.

Button

To read the state of a button connected to GPIO2, you can use this code:

```
from gpiozero import Button
from time import sleep

button = Button(2)

while True:
    if button.is_pressed:
        print("Pressed")
    else:
        print("Released")
    sleep(1)
```

Button functionality includes the properties `is_pressed` and `is_held`;
callbacks `when_pressed`, `when_released`, and `when_held`; and
methods `wait_for_press()` and `wait_for_release`.

Button + LED

To connect the LED and button together, you can use this code:

```
from gpiozero import LED, Button

led = LED(17)
button = Button(2)

while True:
    if button.is_pressed:
        led.on()
    else:
        led.off()
```

EXP.NO:8 DATE:	INTERFACING SENSORS 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

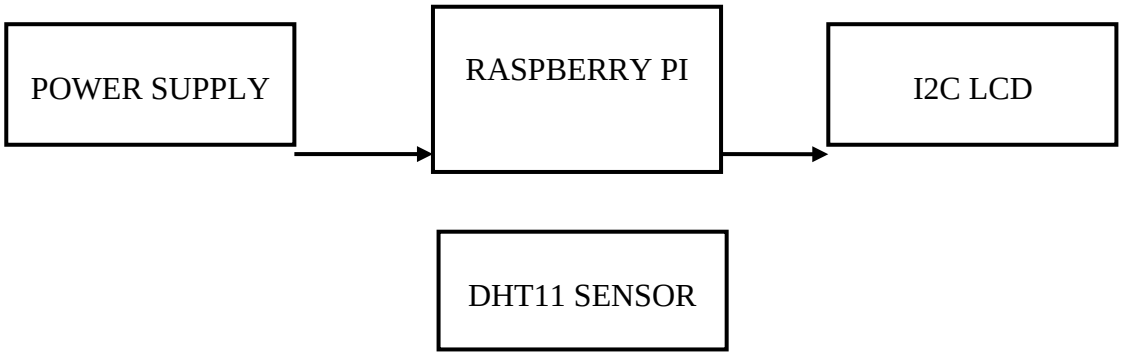
PROGRAM

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

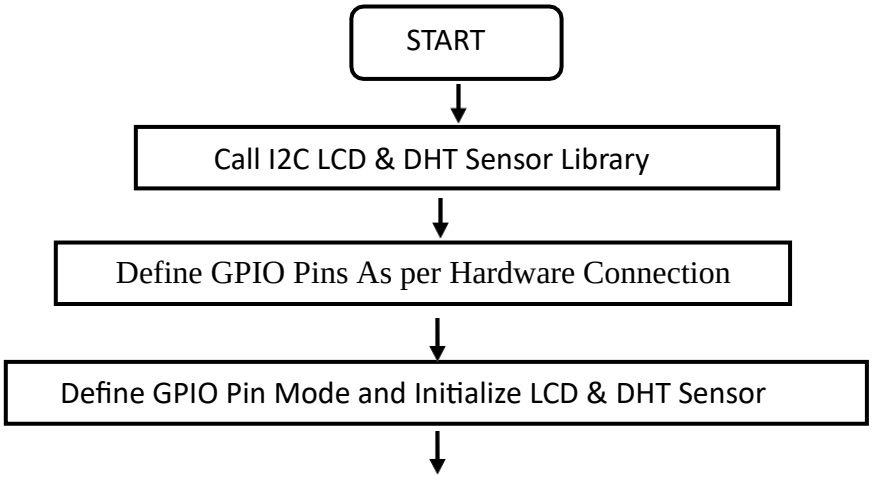
CONNECTION TABLE

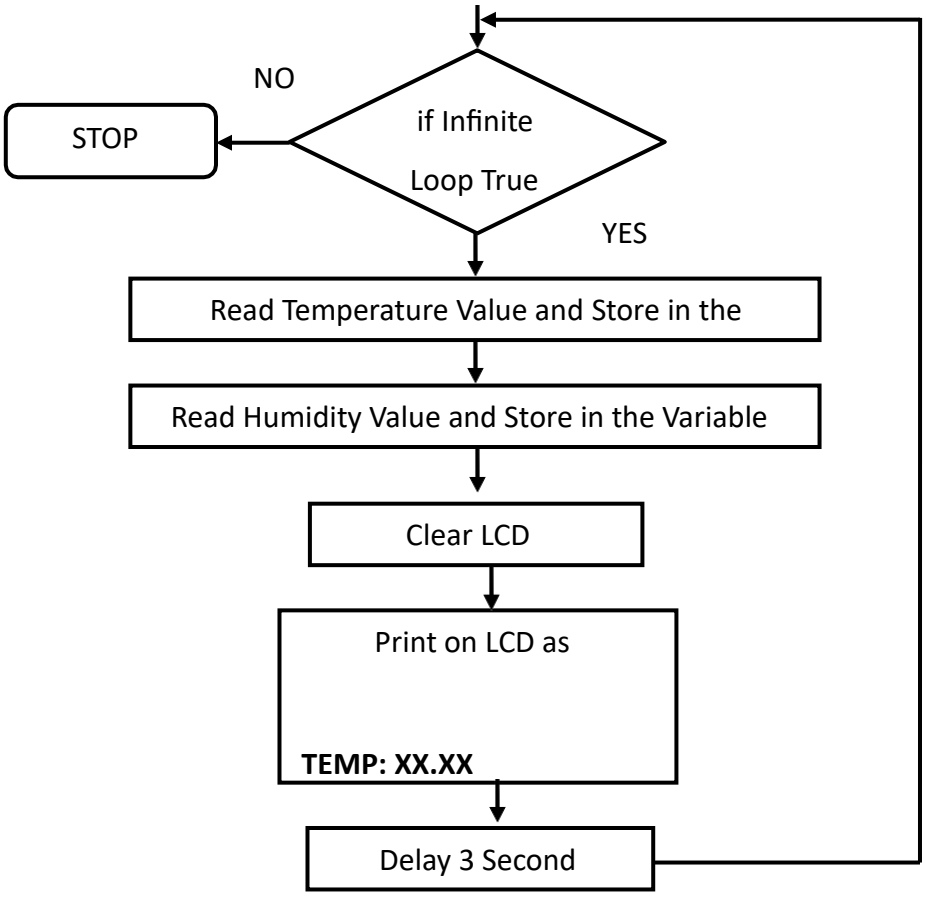
S.NO	CONNECTION		PIN MODE
	COMPONENT	RASPBERRY PI	
1	VCC	5V	POWER +
2	GND	GND	POWER -
3	LCD (SDA)	SPIO 02	Serial Data
4	LCD (SCL)	GPIO 03	Serial Clock
5	DHT11	GPIO 25	Digital Input

BLOCK DIAGRAM

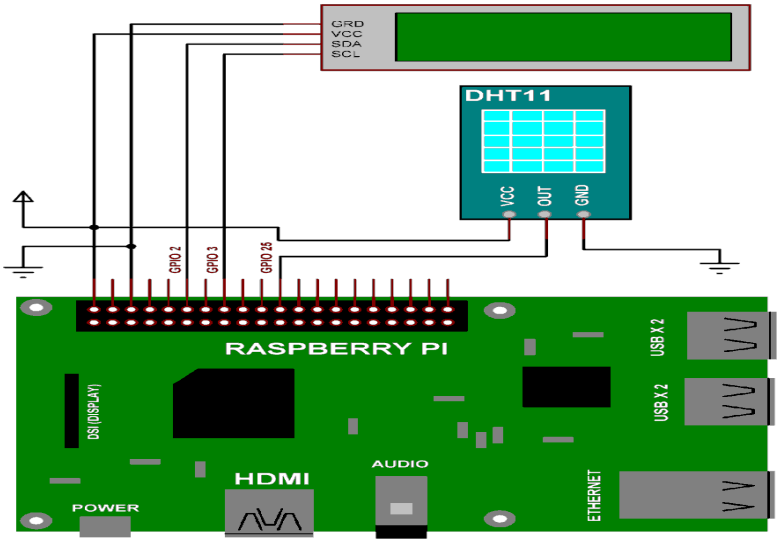


FLOW CHART:





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.

EXP.NO:9 DATE:	COMMUNCATE BETWEEN ARDUINO AND RASPBERRY PI

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

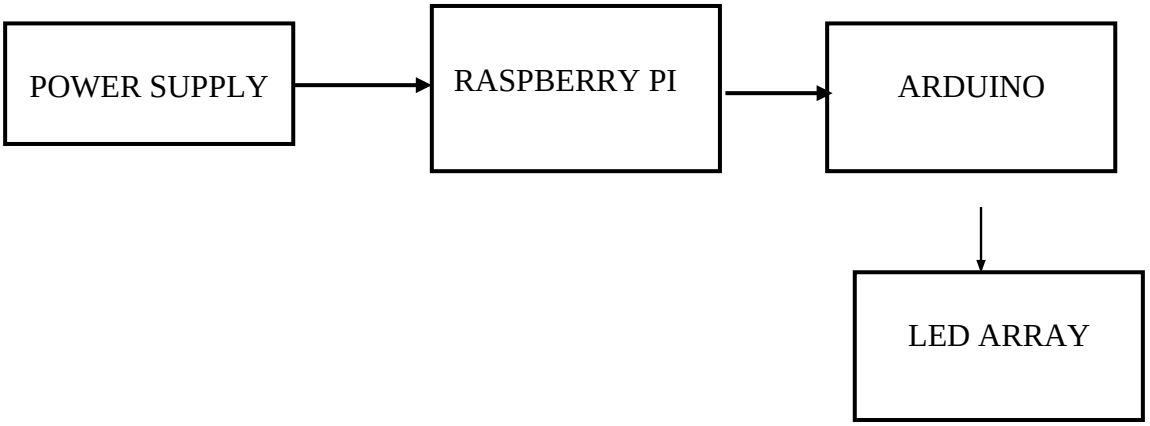
RASPBERRY PI CONNECTION TABLE

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

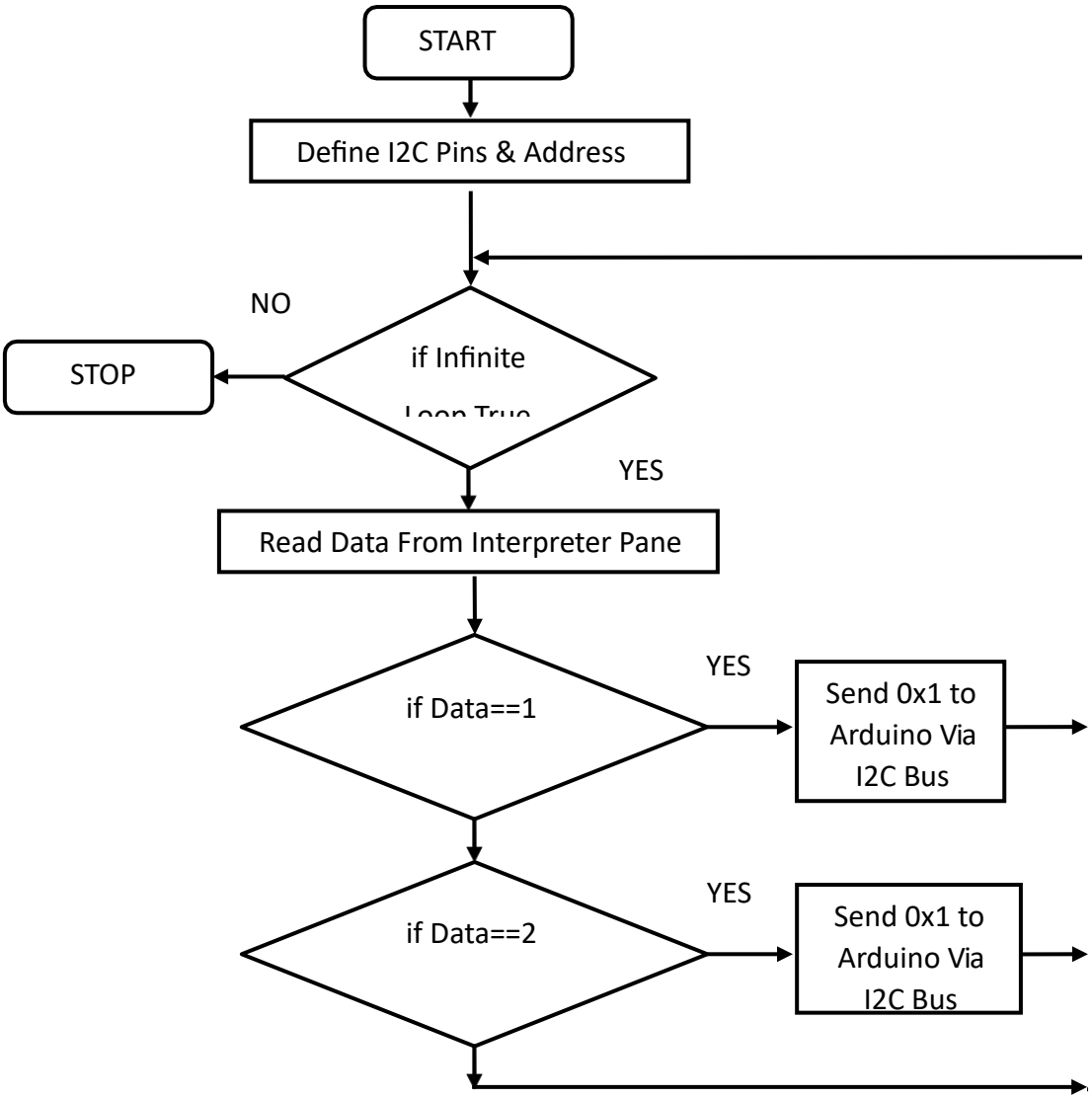
RASPBERRY PI CONNECTION TABLE

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

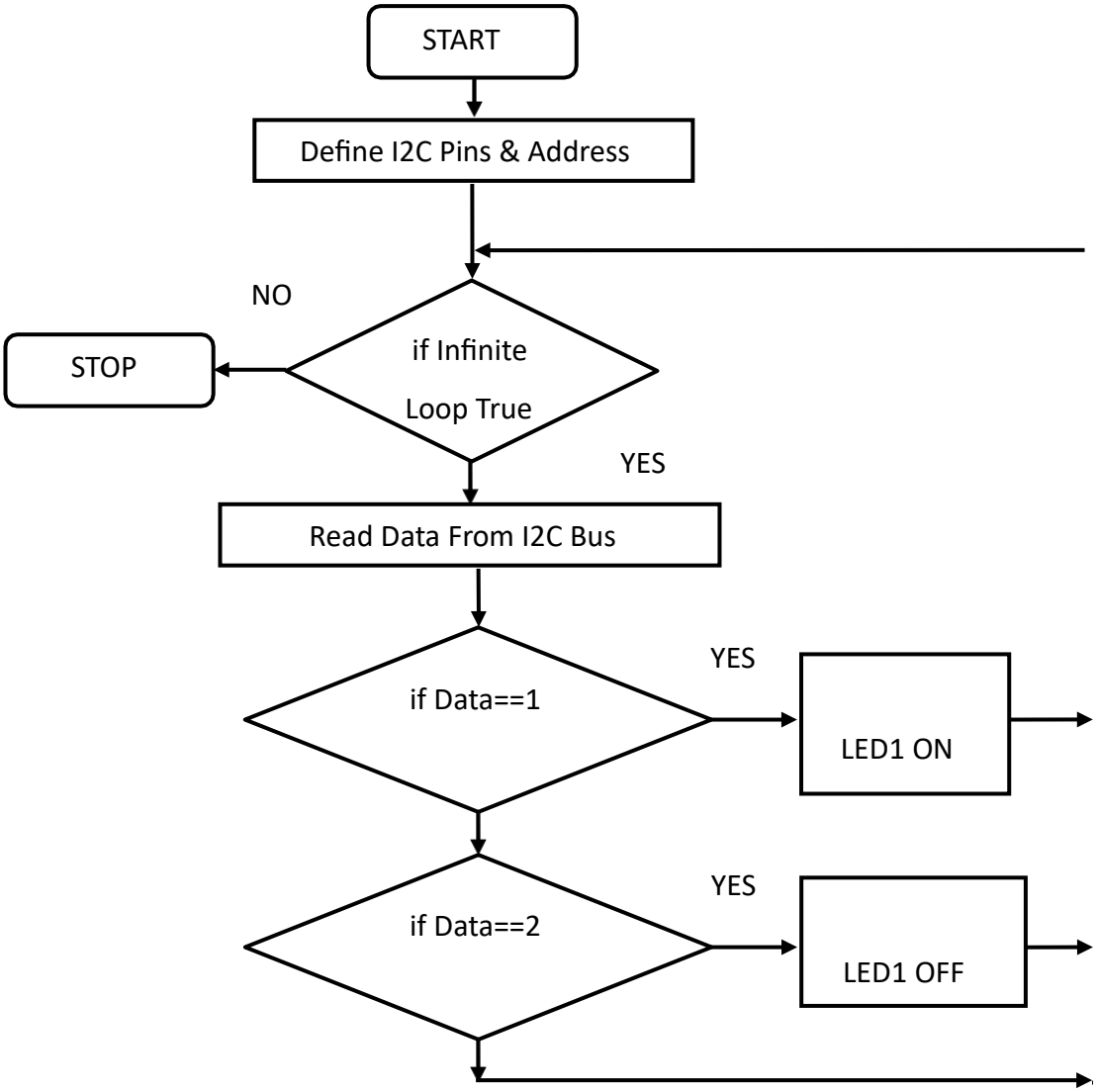
BLOCK DIAGRAM



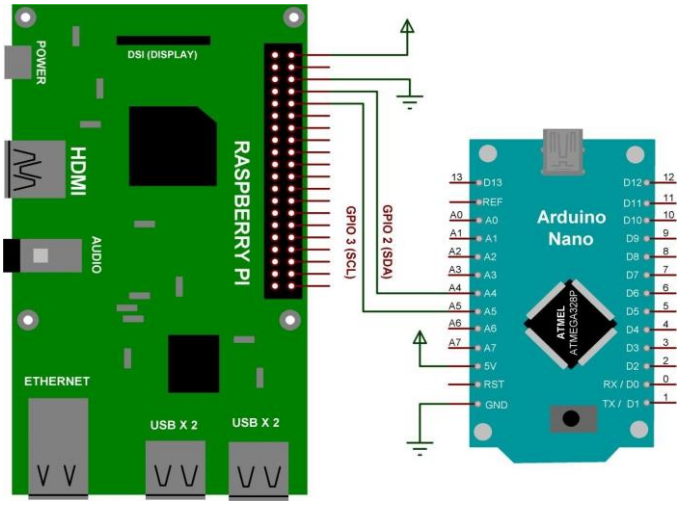
RASPBERRY PI FLOW CHART



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.

EXP.NO:10 DATE:	SETUP A CLOUD PLATFORM TOLOG 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

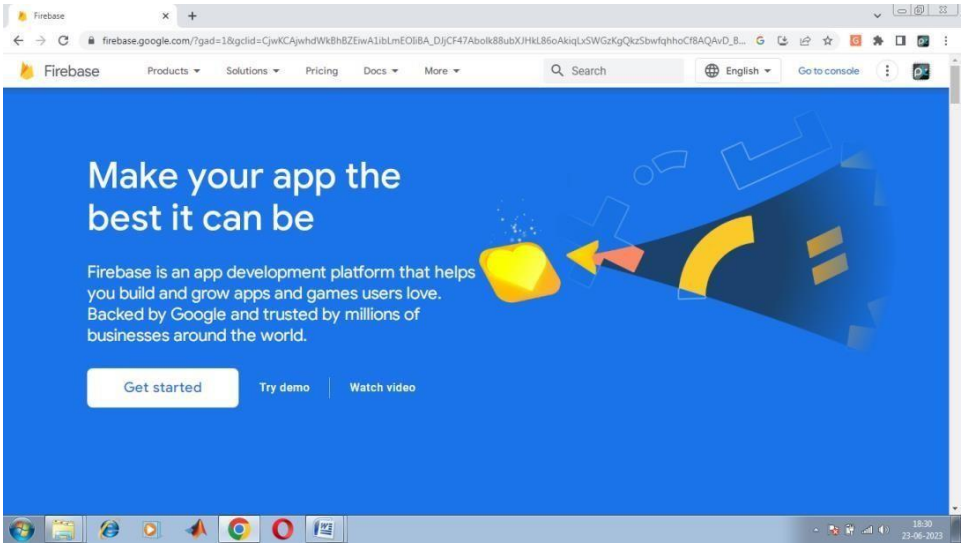
STEP_1:

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

STEP_2:

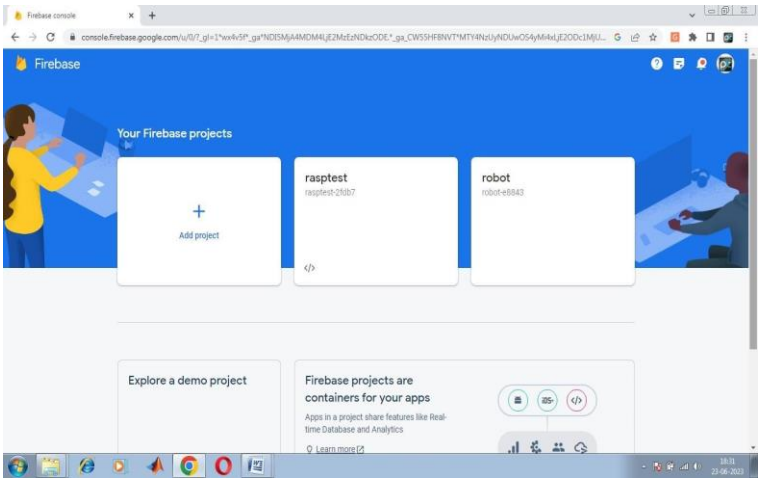
- Login Using Your Google Account - If You Are Not Already Logged In.

STEP_3:



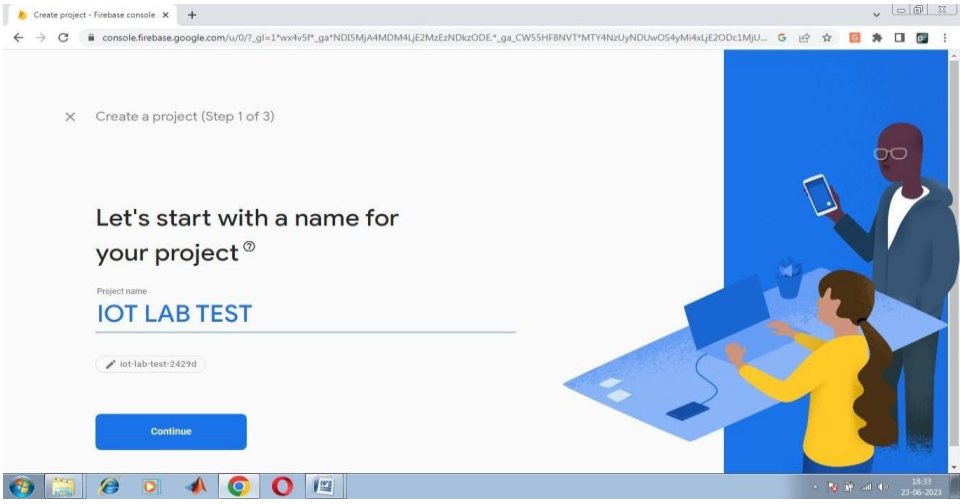
- Click **Get Started**

STEP_4:



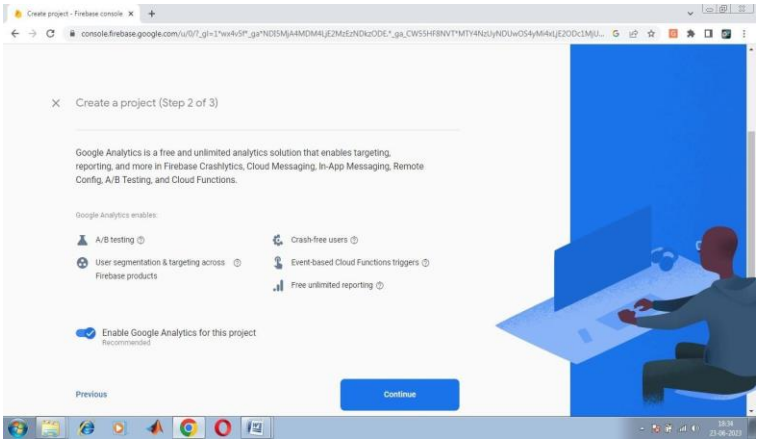
- Click **Add Prjoject**

STEP_5:



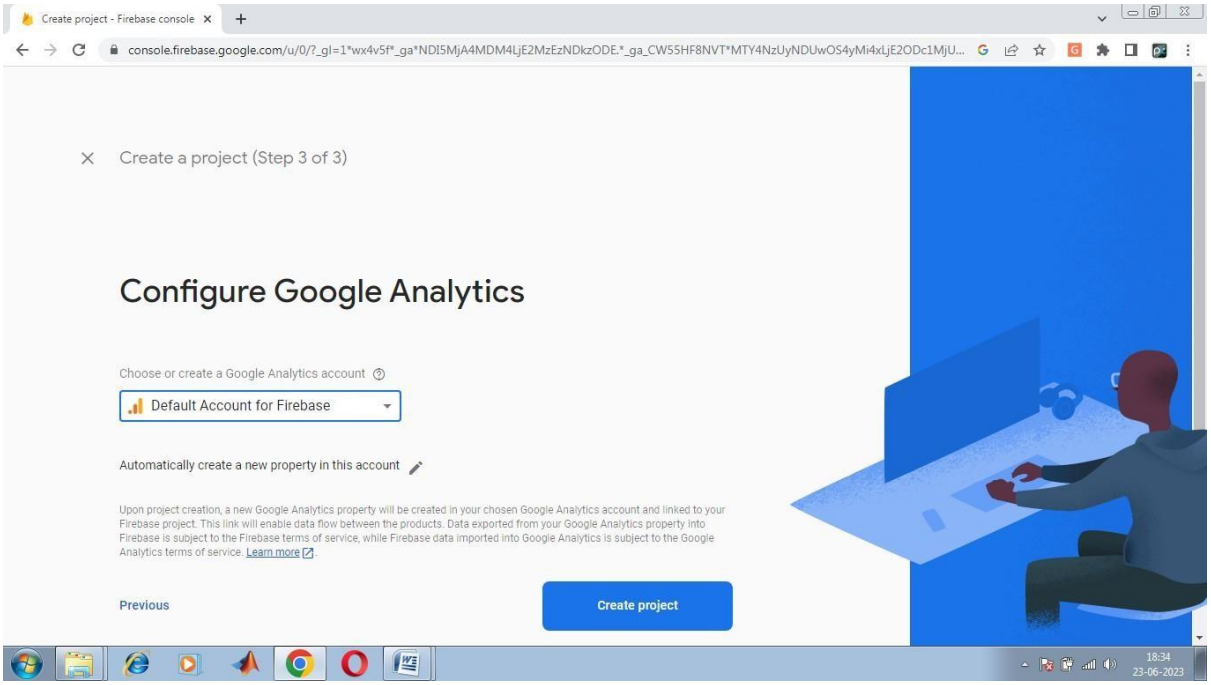
- Enter Your **Project Name and** Click **Continue**

STEP_6:



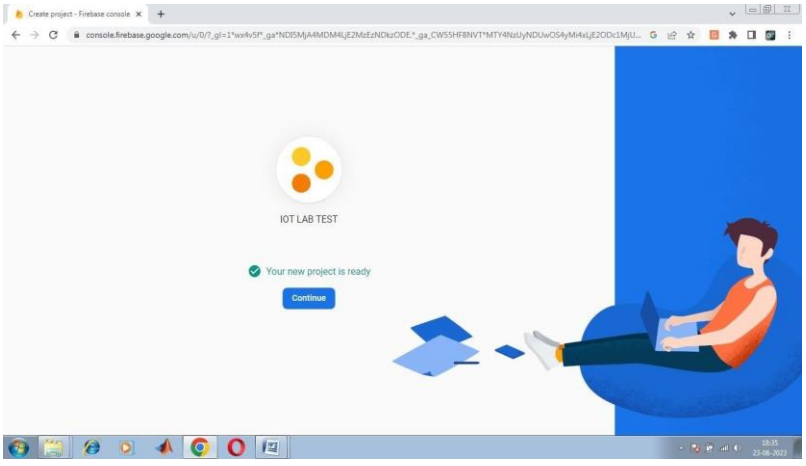
Click **Continue**

- **STEP_7:**



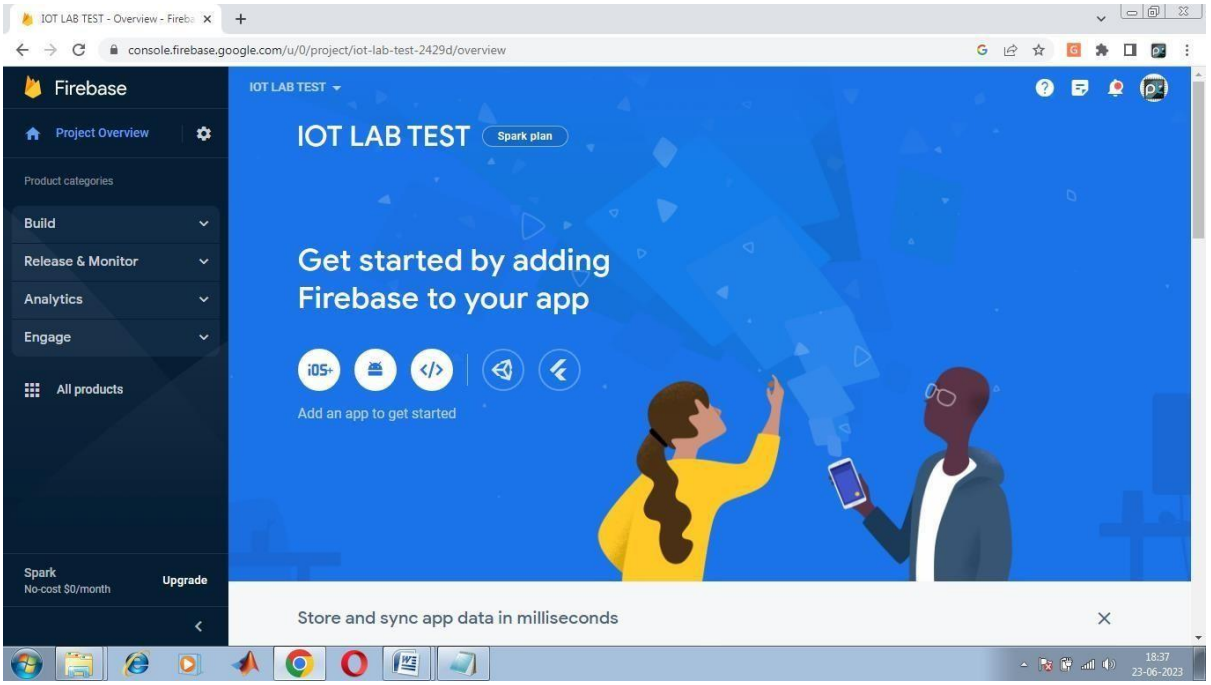
- Select **Default Account For Firebase** And Click **Create Project**

STEP_8:



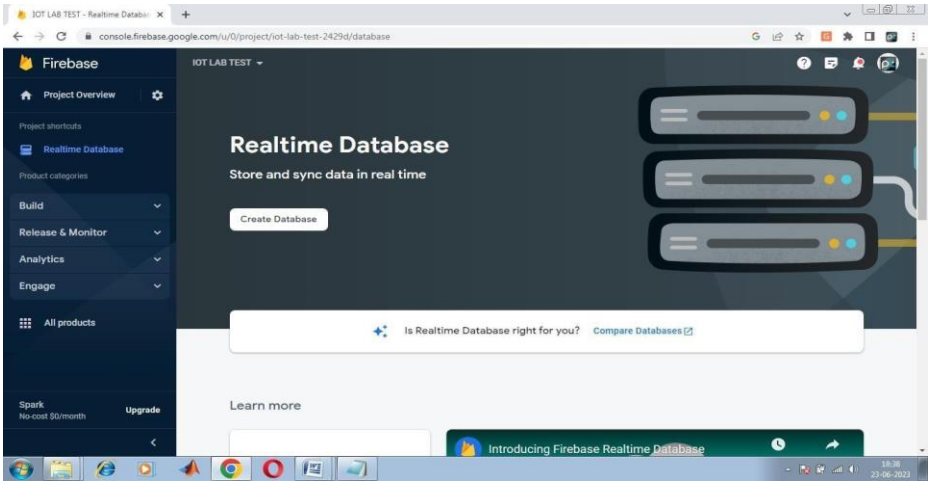
Project Is Created And Click **Continue**

- **STEP_9:**



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

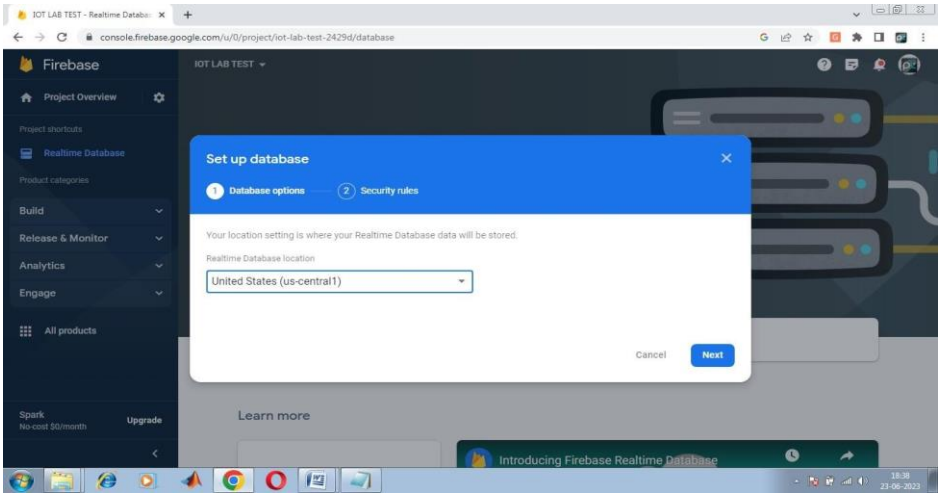
STEP_10:



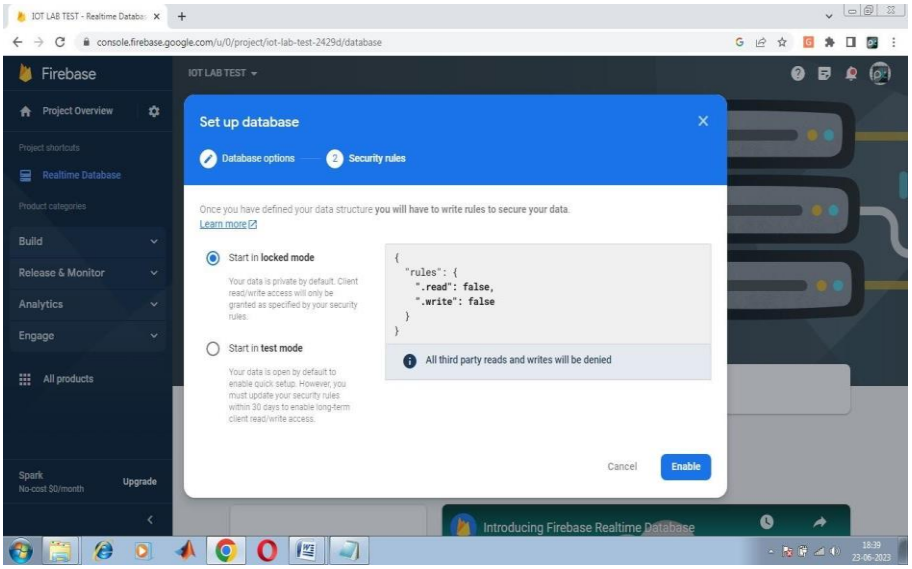
Click **Create Database**

- **STEP_11:**

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

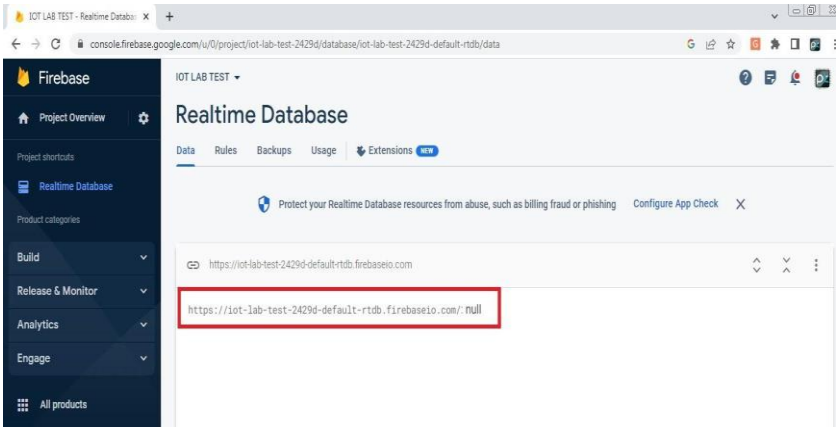


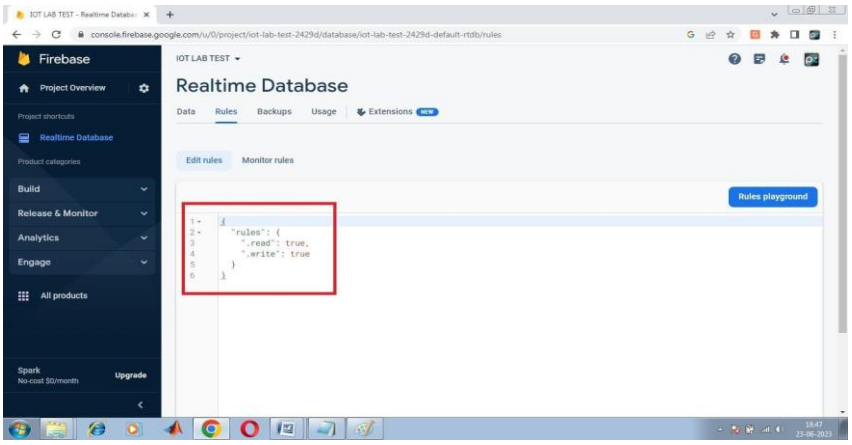
STEP_12:



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

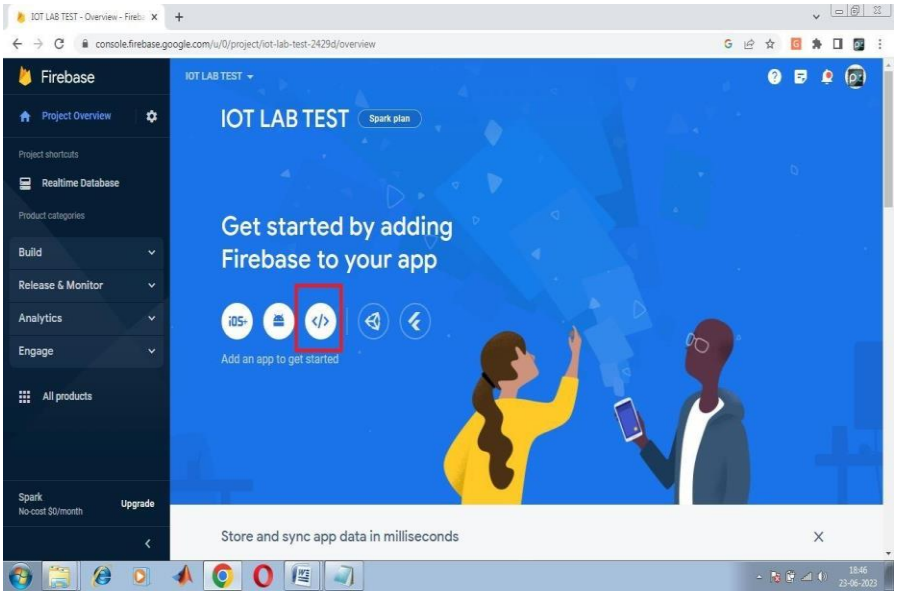
STEP_13:





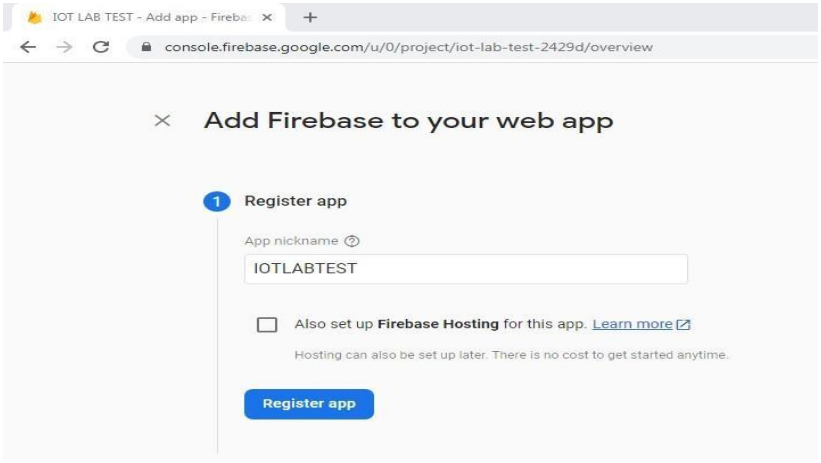
- Now **Real Time Database Is Created** With Host URL.Click Rules And Change **Write And Read Rules As true** And Click **Publish**

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**

EXP.NO:11 DATE:	LOG THE DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM

AIM:

To Design and Implement IOT Based Weather Station ,log the data and upload it to the cloud platform 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**

7. else Update Climate Status on IoT Firebase Database

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

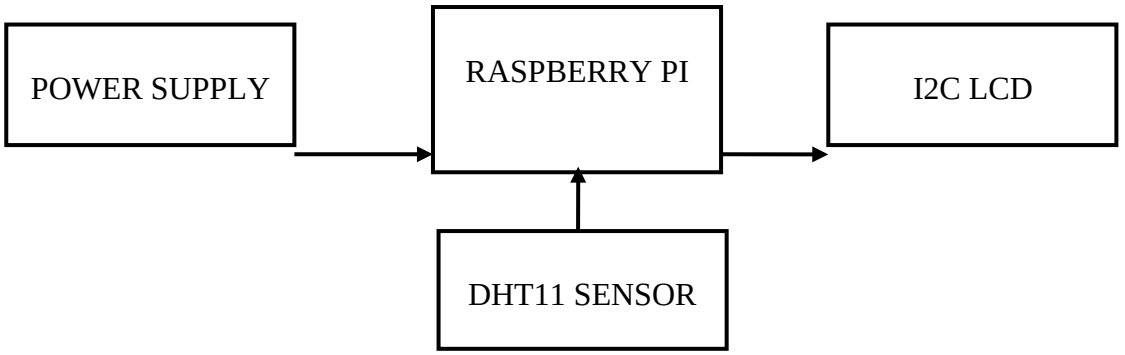
2fdb7.firebaseapp.com", "databaseURL": "https://rasptest-2fdb7-default-rtdb.firebaseio.com/", "storageBucket": "rasptest123" }	
firebase = pyrebase.initialize_app(config)	Initialize Firebase
GPIO.setmode(GPIO.BCM)	Define as Physical numbering
GPIO.setwarnings(False)	Warning Off for GPIO pins
lcd.lcd_clear()	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

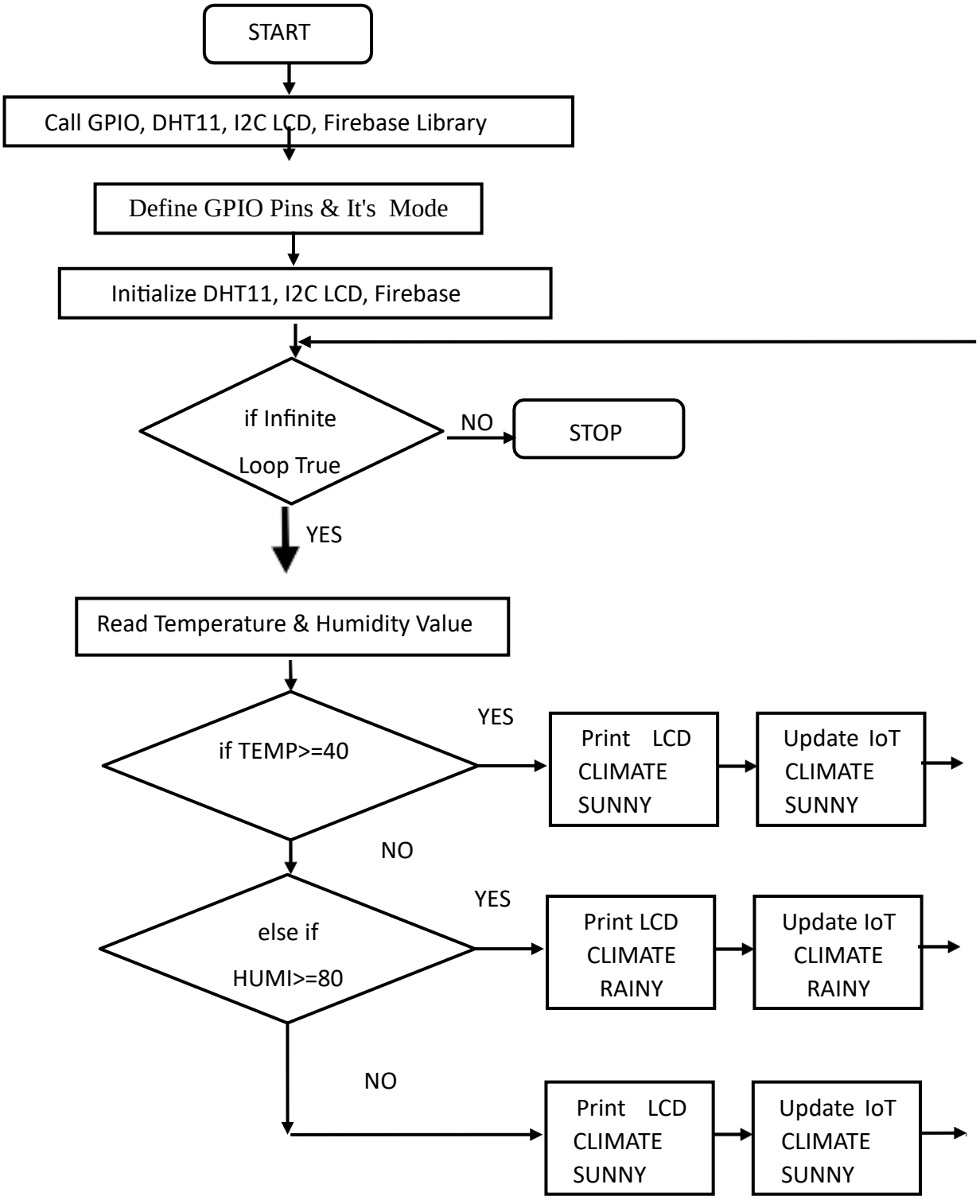
CONNECTION TABLE

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

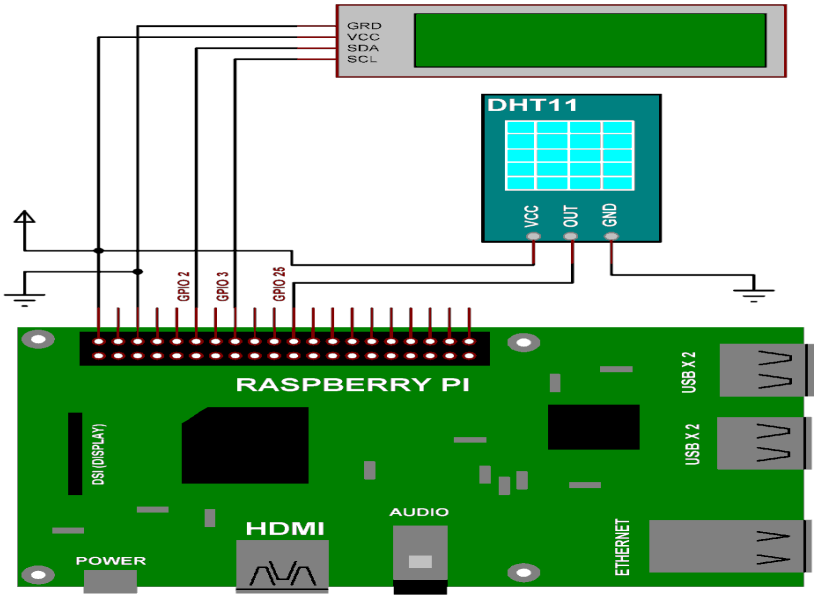
BLOCK DIAGRAM



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



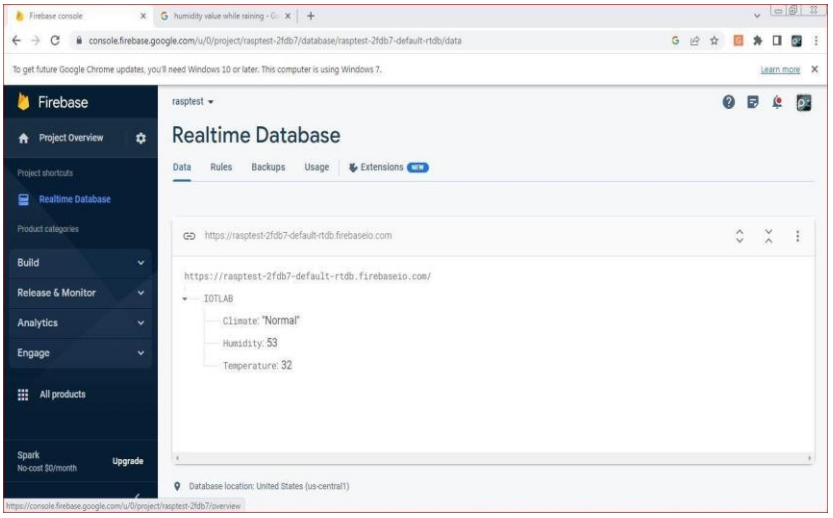
TERMINAL COMMAND LINE

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



EXP.NO:12 DATE:	DESIGN AN IOT BASED SYSTEM

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

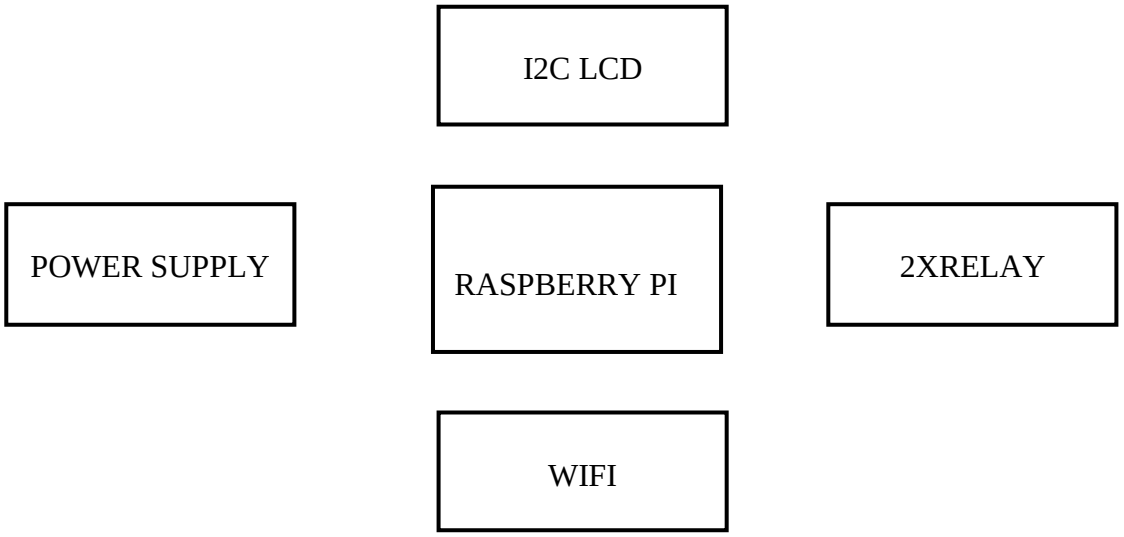
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" IoT Based ", 1)	Print **IoT Based** on LCD 1st Line
lcd.lcd_display_string("Home Automation" , 2)	Print **Home Automation** on LCD 2nd Line
time.sleep(2)	Delay 2 Second
lcd.lcd_clear()	Clear LCD
lcd.lcd_display_string(" System", 1)	Print **System** on 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

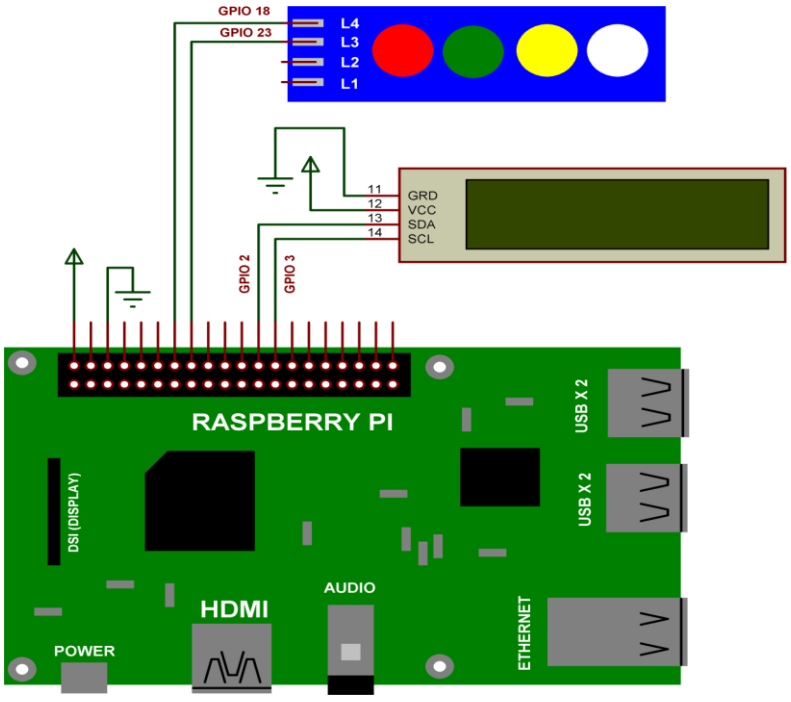
CONNECTION TABLE

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

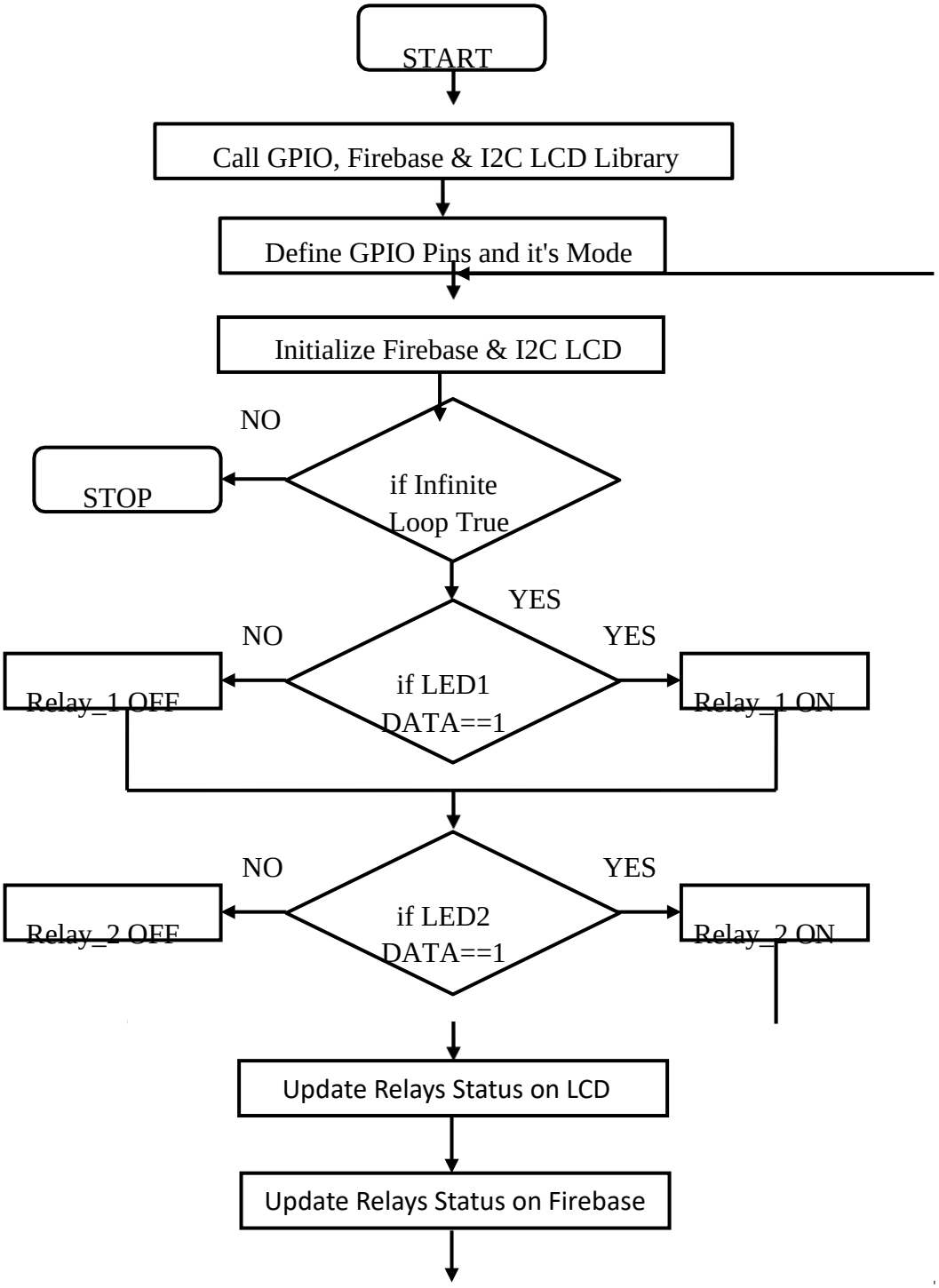
BLOCK DIAGRAM



CIRCUIT DIAGRAM



FLOW CHART


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

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