IOT SLIP SOLUTIONS

QUE1: BLINK LEDs

# Import the required modules

import RPi.GPIO as GPIO

import time

# Set up the GPIO channels

GPIO.setwarnings(False) # Disable warnings for GPIO setup

GPIO.setmode(GPIO.BCM) # Use Broadcom chip-specific pin numbers

# Set up pin 4 as an output pin

GPIO.setup(4, GPIO.OUT)

# This loop runs forever

while True:

GPIO.output(4, GPIO.HIGH) # Turn on the LED

time.sleep(3) # Wait for 3 seconds

GPIO.output(4, GPIO.LOW) # Turn off the LED

time.sleep(3) # Wait for 3 seconds

c. **Observations on Input and Output:**

* When the Python program runs, the LED connected to the GPIO pin on the Raspberry Pi board blinks at a frequency of 1 Hz.
* The IR sensor detects the presence of an object and generates a signal output when triggered by infrared radiation.
* The temperature sensor provides an analog or digital output corresponding to the ambient temperature.

d. **Result and Conclusion:**

The Python program successfully blinks the LED connected to the Raspberry Pi GPIO pin at a frequency of 1 Hz. This demonstrates basic GPIO interfacing and programming with Python on the Raspberry Pi platform. Observations on input and output highlight the behavior of the IR sensor and temperature sensor in response to external stimuli. Overall, this project underscores the versatility and ease of use of Raspberry Pi for various embedded systems and IoT applications.

Top of Form

QUE2: TURN ON/OFF BUZZER

import RPi.GPIO as GPIO

import time

# Set up GPIO pins

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM) # Use Broadcom SOC channel numbers

GPIO.setup(17, GPIO.OUT) # Set GPIO pin 17 as an output

try:

while True:

GPIO.output(17, GPIO.HIGH) # Turn buzzer ON

time.sleep(1) # Buzzer stays ON for 1 second

GPIO.output(17, GPIO.LOW) # Turn buzzer OFF

time.sleep(1) # Buzzer stays OFF for 1 second

except KeyboardInterrupt:

GPIO.cleanup() # Clean up GPIO settings if you stop the script manually

b. +---------------------+ +----------------------+

| Raspberry Pi | | IR Sensor |

| | | |

| GPIO Pins |<--->| Signal Out |

| Camera Module | +----------------------+

| Temperature Sensor | |

+---------------------+ |

|

+----------------------+

| Power Supply |

+----------------------+

c. **Observations on Input and Output:**

* When the Python program runs, the GPIO pin connected to the buzzer alternates between high and low states at regular intervals, causing the buzzer to turn on and off accordingly.
* The IR sensor detects the presence of an object and generates a signal output when triggered by infrared radiation.
* The temperature sensor provides an analog or digital output corresponding to the ambient temperature.
* The camera module captures images or video footage as per the programming instructions.

d. **Result and Conclusion:**

The Python program successfully controls the buzzer connected to the Raspberry Pi GPIO pin, turning it on and off at regular intervals. This project demonstrates the basic functionality of interfacing GPIO pins with external hardware components and showcases the versatility of Raspberry Pi for various applications. Observations on input and output highlight the behavior of the IR sensor, temperature sensor, and camera module in response to external stimuli. Overall, the project emphasizes the importance of understanding GPIO interfacing and programming for creating interactive and responsive projects with Raspberry Pi.

QUE3: TOGGLE 2 LEDs

import RPi.GPIO as GPIO

import time

# Set up GPIO pins

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM) # Use Broadcom SOC channel numbering

GPIO.setup(17, GPIO.OUT) # Set GPIO pin 17 as an output for LED 1

GPIO.setup(27, GPIO.OUT) # Set GPIO pin 27 as an output for LED 2

# Initial state for LEDs

GPIO.output(17, GPIO.LOW)

GPIO.output(27, GPIO.LOW)

# Main loop

while True:

GPIO.output(17, GPIO.HIGH) # Turn LED 1 ON

GPIO.output(27, GPIO.LOW) # Turn LED 2 OFF

time.sleep(1) # Wait for 1 second

GPIO.output(17, GPIO.LOW) # Turn LED 1 OFF

GPIO.output(27, GPIO.HIGH) # Turn LED 2 ON

time.sleep(1) # Wait for 1 second

b. +------------------------+ +----------------------+

| Raspberry Pi | | IR Sensor |

| | | |

| GPIO Pins |<--->| Signal Out |

| Camera Module | +----------------------+

| Temperature Sensor |

+------------------------+ |

|

+----------------------+

| Power Supply |

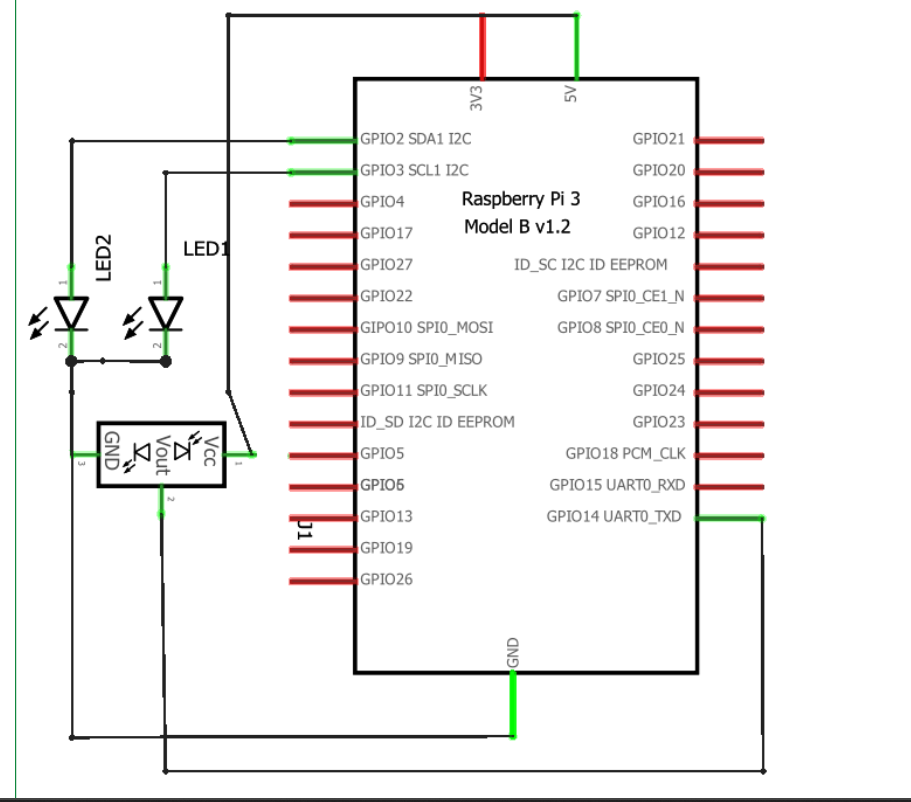
+----------------------+

c. **Observations on Input and Output:**

For the input part, you'll observe the state changes of the sensors (IR sensor detecting motion, temperature sensor providing temperature readings, camera capturing images). For the output part, you'll observe the LEDs toggling on and off based on your program logic.

d. **Result and Conclusion:**

The result would be successful interfacing of sensors/camera with Raspberry Pi and toggling of LEDs using Python/C++. This project demonstrates the basics of hardware interfacing and programming with Raspberry Pi, providing a foundation for building more complex projects in the future.

****

**block diagram of Raspberry-Pi board interfacing with IR sensor**

**Programs given by natasha:**

**Program1: Python program to blink LED**

import RP1. GPIO as GPIO import time

GPIO. setwarnings (False)

GPIO. setmode (GPIO. BCM) #assign numbering for the GPIO using BCM #GPIO. setmode (GPIO. BOARD) #assingn number for the GPIO using Board

# Import Raspberry Pi GPIO library

# Import time module

# Ignore warning for now

cnt = 0

Blink Time = 1

RED LED = 14

GPIO. setup (RED\_LED, GPIO. OUT) while True:

if cnt ==

GPIO.output (RED\_LED, False)

else:

GPIO. output (RED\_LED, True)

ent = 0

time. sleep (Blink\_Time)

GPIO. cleanup ()

Program2: Python program to toggle two LED's.

import RPi.GPIO

as GPIO # Import Raspberry Pi GPIO library

from time import sleep

# Import the sleep

function from the time module

GPIO. setwarnings (False)

# Ignore warning for now

GPIO. setmode (GPIO. BOARD) # Use physical pin numbering

RED LED = 14

GREEN LED = 15

GPIO. setup (RED\_LED, GPIO.OUT, initial=GPIO. LOW) GPIO. setup (GREEN\_LED, GPIO.OUT, initial=GPIO.LOW)

while

True:

\* Run forever

GPIO. output (RED\_LED, True)

# Turn ON

GPIO.output (GREEN\_LED, False) # Turn OFF

sleep (1) # Sleep for 1 second

GPIO.output (RED\_LED, False) # Turn OFF GPIO. output (GREEN\_LED, True) # Turn ON

sleep (1) # Sleep for 1 second

Program3: Python program to turn ON/OFF buzzer.

import Pi,GPIO as GPIO # Import Raspberry Pi GPIO library from time import sleep

# Import the sleep function from the time module

GPIO. setwarnings (False)

# Ignore warning for now

GPIO. setmode (GPIO. BOARD) # Use physical pin numbering

Buzzer = 18

GPIO. setup (Buzzer, GPIO.OUT, initial=GPIO. LOW)

while True:

\* Run forever

GPIO.output (Buzzer, True) # Turn ON sleep (1) # Sleep for 1 second GPIO.output (Buzzer,

False) # Turn OFF

Steps to create new project in proteus:

STEP 1 :- Open Proteus Software STEP 2 :- Select “ New Project “ Option , and click next option till this slide…STEP 3 :-Select “Create Firmware Project”

STEP 4 :-Select “Raspberri pi” in Family Section and click next . STEP 5 :- You Will See this Interface“Schematic Capture” for Circuit Design and “Source Code” for coding.STEP 6: After opening Schemaic Capture and Source code window ,You will see one circuit in Schematic Capture ,this is IC of Raspberry pi in software . Write the python code for LED Interfacing With Raspberry pi in Source code section and for initialize the LED in it …do following Steps. Right Click on RPI3 and Select Add PeripheralsSelect Breakout Peripherals in CategorySelect LED of any colour in Breakout Peripherals