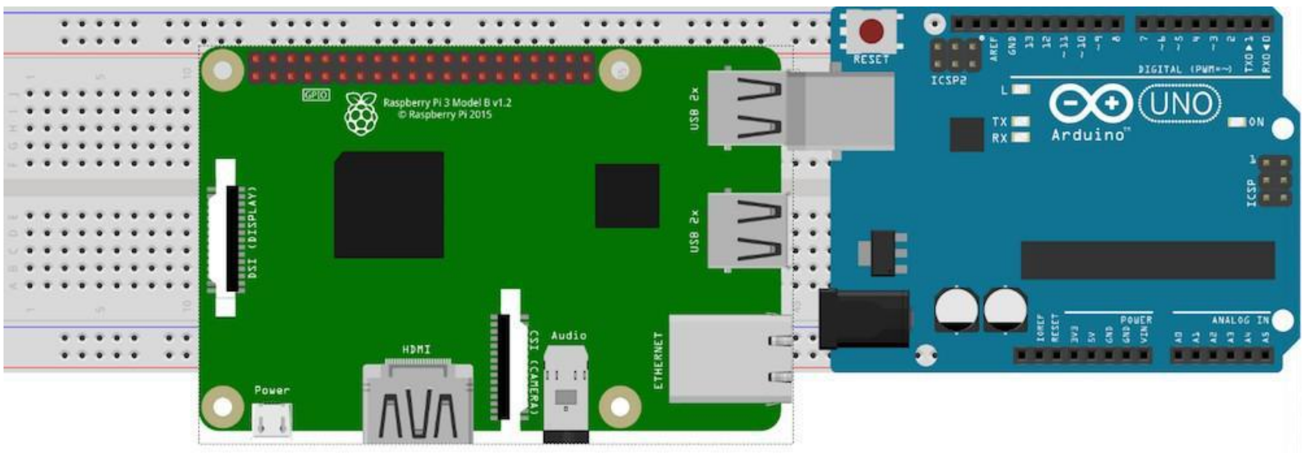
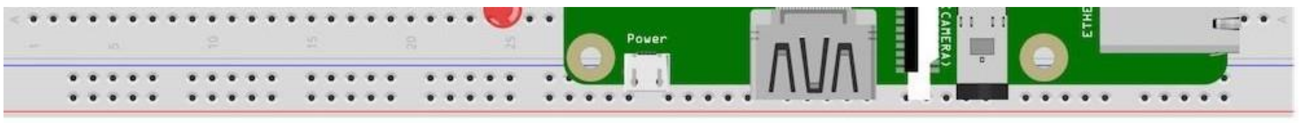
4.3.1 Interface a Raspberry Pi with an Arduino



Interface a Raspberry Pi with an Arduino so the two boards can communicate with one another. Sometimes you may need to connect an Arduino to a Raspberry Pi. For example, if you have sensors, motors, and actuators, you can connect these to the Arduino and make the Arduino send values to and from the Raspberry Pi. This way, we can separate the computing intensive tasks (done by the Raspberry Pi) and controlling tasks (done by the Arduino).

we will connect an Arduino to a Raspberry Pi and have the Arduino send “Hello from Arduino” to the Raspberry Pi, and the Raspberry Pi will blink an LED upon receiving the command from the Arduino.

For communication, we will use simple serial communication over USB cable.So, let’s get started! Connect the LED to pin number 11 as shown in the picture below.



Turn on the Raspberry Pi and open Python 3 in a new window.

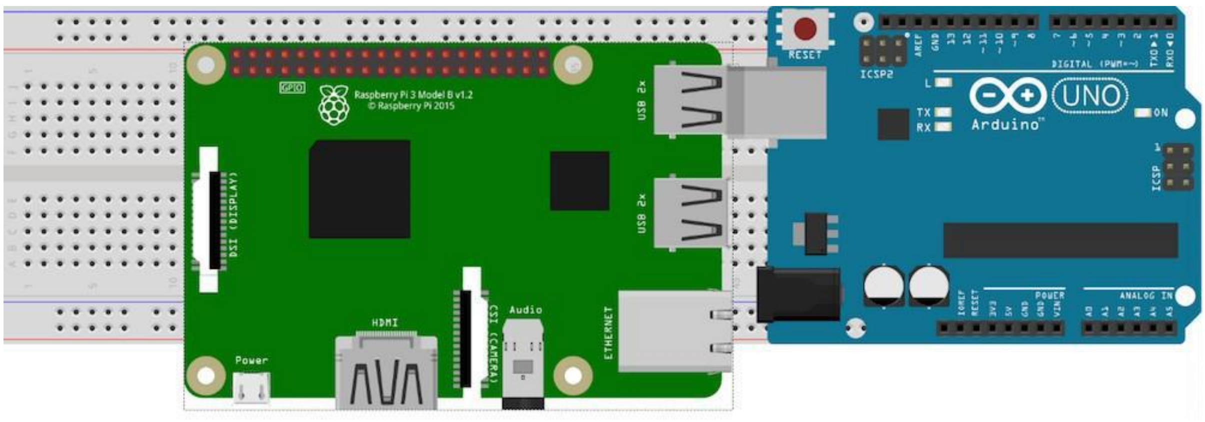
Write the following code in the new window and save it. (Save to your desktop so you don't loseit.)

```
import serial
import RPi.GPIO as GPIOimport time

ser=serial.Serial("/dev/ttyACM0",9600)  #change ACM number as found from ls /dev/tty/ACM*
ser.baudrate=9600
def blink(pin):
GPIO.output(pin,GPIO.HIGH)time.sleep(1)
GPIO.output(pin,GPIO.LOW)time.sleep(1)
return
GPIO.setmode(GPIO.BOARD)GPIO.setup(11, GPIO.OUT)
while True:
read_ser=ser.readline()print(read_ser)
if(read_ser=="Hello From Arduino!"):
blink(11)
```

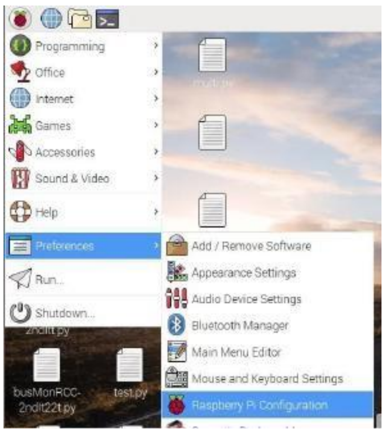
Now open Arduino IDE and upload the following code to your Arduino.String data="Hello From Arduino!";

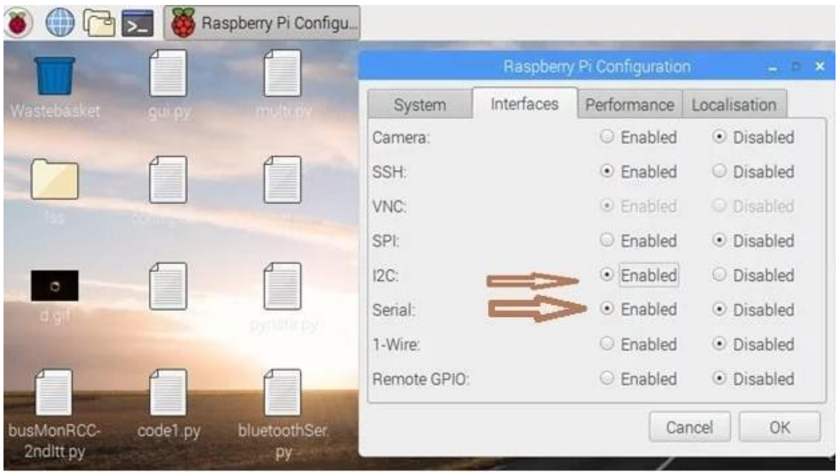
```
void setup()
{
// put your setup code here, to run once:Serial.begin(9600);
}
void loop()
{
// put your main code here, to run repeatedly:Serial.println(data);//data that is being Sent delay(200);
}
```



Make sure the code is uploaded to Arduino.

In your Raspberry Pi interface, be sure to enable Serial and I2C in PiConfig.



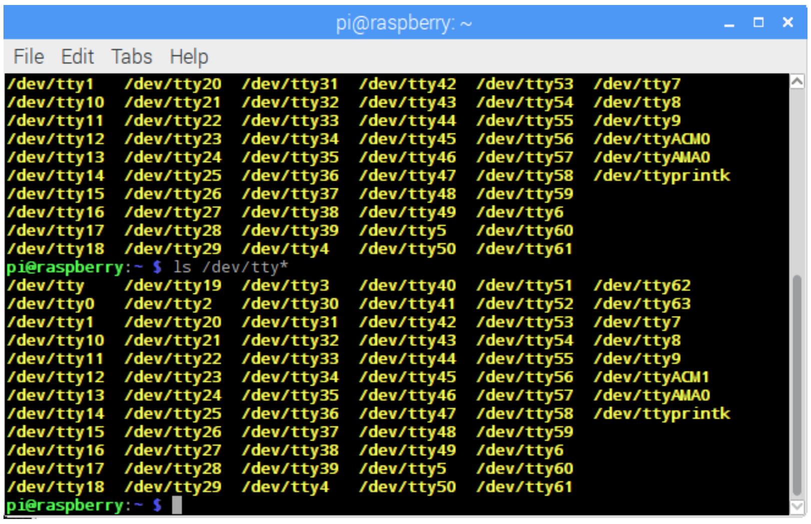


Camera: Enabled [✓ Disabled]
SSH: [✓ Enabled] Disabled
VNC: [✓ Enabled] Disabled
SPI: Enabled [✓ Disabled]
I2C: [✓ Enabled] Disabled
Serial: [✓ Enabled] [✓ Disabled]
1-Wire: Enabled [✓ Disabled]
Remote GPIO: Enabled [✓ Disabled]

Next, you'll need to restart your Raspberry Pi. Open the Terminal and execute these commands:

sudo apt-get install python-serial
sudo pip install pyserial
Connect your Arduino to your Raspberry Pi.Execute.
ls /dev/tty*

Then find a line with /dev/ttyACM0 or something like /dev/ttyACM1 etc. (check for an ACMwith any number 0,1,2 etc.)



```
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyACM0
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyAMA0
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyprintk
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
pi@raspberry:~ $ ls /dev/tty*
/dev/tty    /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0   /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyACM1
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyAMA0
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyprintk
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
pi@raspberry:~ $
```

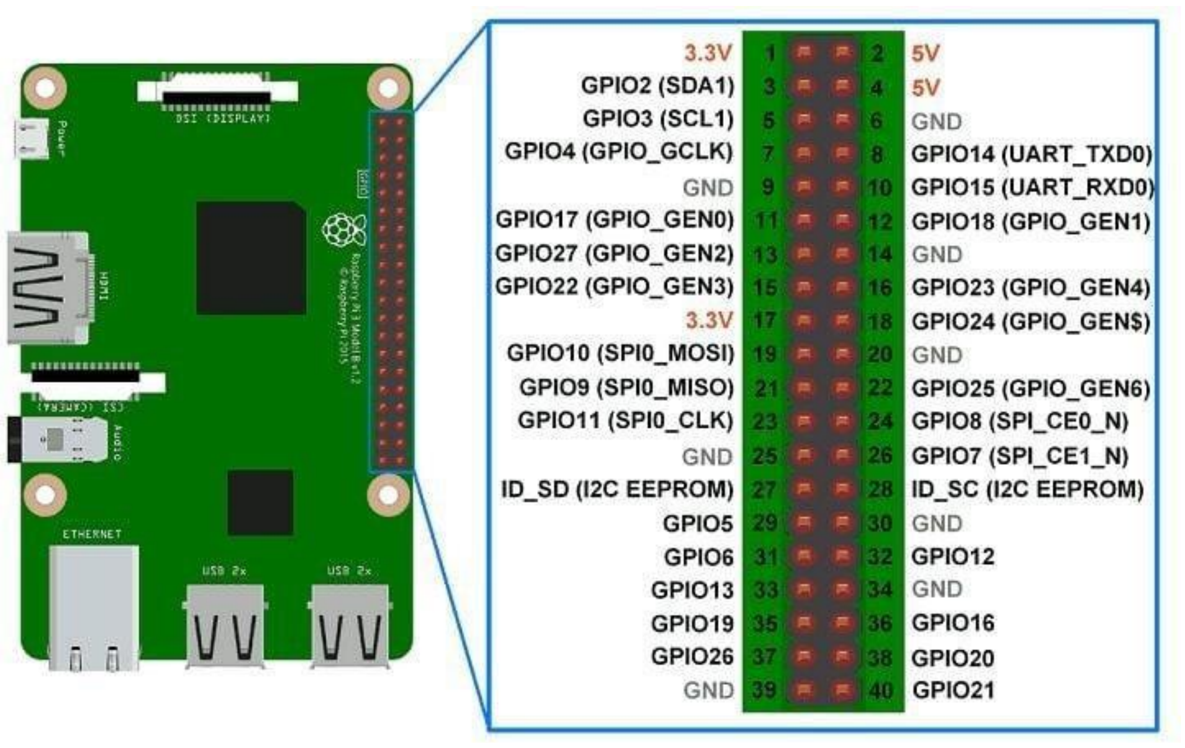
Open Python again and change ser=serial.Serial(“dev/ttyACM1”,9600) to the ACM number you found. So, if in your case you got ACM0, the line should look like this: ser=serial.Serial(“dev/ttyACM0”,9600). Now run the program you just created in Python3. You will see “Hello From Arduino!” in the Python terminal, and your LED should be blinking as well!

4.3.2 Raspberry Pi GPIO Access

GPIO (General Purpose Input Output) pins can be used as input or output and allows raspberry pito connect with general purpose I/O devices.

- Raspberry pi 3 model B took out 26 GPIO pins on board.
- Raspberry pi can control many external I/O devices using these GPIO's.
- These pins are a physical interface between the Pi and the outside world.
- We can program these pins according to our needs to interact with external devices. For example, if we want to read the state of a physical switch, we can configure any of the available GPIO pins as input and read the switch status to make decisions. We can also configure any GPIO pin as an output to control LED ON/OFF.
- Raspberry Pi can connect to the Internet using on-board Wi-Fi or Wi-Fi USB adapter. Once the Raspberry Pi is connected to the Internet then we can control devices, which are connected to the Raspberry Pi, remotely.

GPIO Pins of Raspberry Pi 3 are shown in below figure:



DSI (DISPLAY)
Power
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x
GPIO
3.3V 1 2 5V
GPIO2 (SDA1) 3 4 5V
GPIO3 (SCL1) 5 6 GND
GPIO4 (GPIO_GCLK) 7 8 GPIO14 (UART_TXD0)
GND 9 10 GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0) 11 12 GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2) 13 14 GND
GPIO22 (GPIO_GEN3) 15 16 GPIO23 (GPIO_GEN4)
3.3V 17 18 GPIO24 (GPIO_GEN$)
GPIO10 (SPI0_MOSI) 19 20 GND
GPIO9 (SPI0_MISO) 21 22 GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK) 23 24 GPIO8 (SPI_CE0_N)
GND 25 26 GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM) 27 28 ID_SC (I2C EEPROM)
GPIO5 29 30 GND
GPIO6 31 32 GPIO12
GPIO13 33 34 GND
GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20
GND 39 40 GPIO21

Raspberry Pi 3 Model B GPIO Pin Mapping

Some of the GPIO pins are multiplexed like I2C, SPI, UART etc. We can use any of the GPIO pins for our application.

Pin Numbering

We should define GPIO pin which we want to use as an output or input. But Raspberry Pi hastwo ways of defining pin number which are as follows:

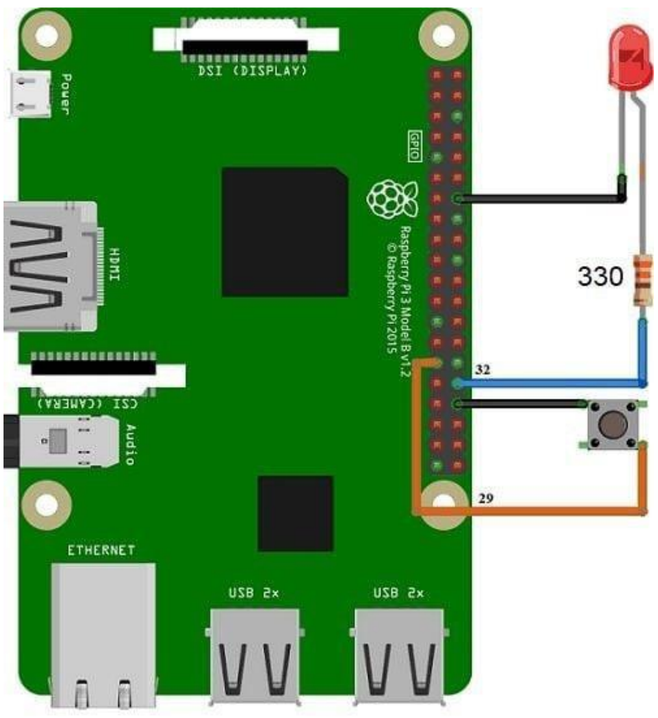
- **GPIO Numbering**
- **Physical Numbering**

In **GPIO Numbering**, pin number refers to number on Broadcom SoC (System on Chip). So, weshould always consider the pin mapping for using GPIO pin.

While in **Physical Numbering**, pin number refers to the pin of 40-pin P1 header on Raspberry PiBoard. The above physical numbering is simple as we can count pin number on P1 header and assign it as GPIO.

But, still we should consider the pin configuration diagram shown above to know which areGPIO pins and which are VCC and GND.

Control LED with Push Button using Raspberry Pi



Control LED using Raspberry Pi Interfacing Diagram

Example

Now, let’s control LED using a switch connected to the Raspberry Pi. Here, we are using Pythonand C (WiringPi) for LED ON-OFF control.

Control LED using Python

Now, let’s turn an ON and OFF LED using Python on Raspberry Pi. The switch is used tocontrol the LED ON-OFF.

Python Program for Raspberry Pi to control LED using Push Button

```
import RPi.GPIO as GPIO          #import RPi.GPIO module
LED = 32                         #pin no. as per BOARD, GPIO18 as per BCMSwitch_input = 29
                                 #pin no. as per BOARD, GPIO27 as per BCM
GPIO.setwarnings(False)          #disable warnings GPIO.setmode(GPIO.BOARD)  #set pin
numbering format GPIO.setup(LED, GPIO.OUT)  #set GPIO as output GPIO.setup(Switch_input,
GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True: if(GPIO.input(Switch_input)):
GPIO.output(LED,GPIO.LOW)
else:
GPIO.output(LED,GPIO.HIGH)
```

Functions Used:

RPi.GPIO

To use Raspberry Pi GPIO pins in Python, we need to import RPi.GPIO package which has classto control GPIO. This RPi.GPIO Python package is already installed on Raspbian OS. So, we don’t need to install it externally. Just, we should include library in our program to use functionsfor GPIO access using Python. This is given as follows.

import RPi.GPIO as GPIO

GPIO.setmode (Pin Numbering System)

This function is used to define Pin numbering system i.e. GPIO numbering or Physicalnumbering.

Pin Numbering System = BOARD/BCM

E.g. If we use pin number 40 of P1 header as a GPIO pin which we have to configure as outputthen,

In BCM,

GPIO.setmode(GPIO.BCM)GPIO.setup(21, GPIO.OUT)

In BOARD,

GPIO.setmode(GPIO.BOARD)GPIO.setup(40, GPIO.OUT)

GPIO.setup (channel, direction, initial value, pull up/pull down)

This function is used to set the direction of GPIO pin as an input/output.**channel** – GPIO pin number as per numbering system. **direction** – set direction of GPIO pin as either Input or Output.

initial value – can provide initial value

pull up/pull down – enable pull up or pull down if requiredFew examples are given as follows,

- GPIO as Output
  GPIO.setup(channel, GPIO.OUT)
- GPIO as Input
  GPIO.setup(channel, GPIO.IN)
- GPIO as Output with initial value
  GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH)
- GPIO as Input with Pull up resistor
  GPIO.setup(channel, GPIO.IN, pull_up_down = GPIO.PUD_UP) GPIO.output(channel, state)

This function is used to set the output state of GPIO pin.

channel – GPIO pin number as per numbering system.

state – Output state i.e. HIGH or LOW of GPIO pin

e.g.

GPIO.output(7, GPIO.HIGH)

GPIO.input(channel)

This function is used to read the value of GPIO pin.e.g.

GPIO.input(9)

Control LED using C (WiringPi)

We can access Raspberry Pi GPIO using C. Here, we are using WiringPi library for accessingRaspberry Pi GPIO using C.

Before implementing LED blinking using wiringPi, you can refer How to use WiringPi library.

C (WiringPi) Program for Raspberry Pi to control LED using Push Button

```
#include <wiringPi.h>
#include <stdio.h>

int LED = 26; /* GPIO26 as per wiringPi, GPIO12 as per BCM, pin no.32 */

int switch_input = 21;  /* GPIO21 as per WiringPi, GPIO5 as per BCM, pin no.29 */int main(){

wiringPiSetup();                    /* initialize wiringPi setup */ pinMode(LED,OUTPUT); /* set GPIO

as output */pullUpDnControl(switch_input, PUD_UP);

while (1){

if(digitalRead(switch_input))

digitalWrite(LED,LOW); /* write LOW on GPIO */else


}

}
digitalWrite(LED, HIGH);  /* write HIGH on GPIO */
```