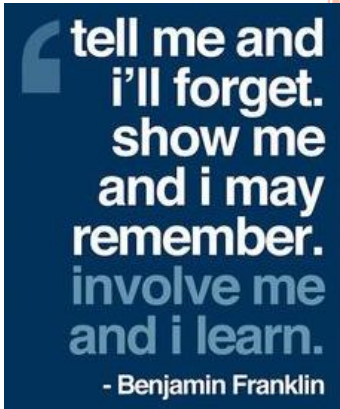
GPIO Programming in Raspberry Pi using C (WiringPi)

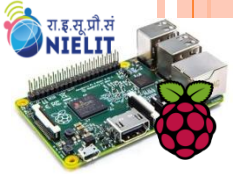
Dr. Sarwan Singh
DeputyDirector(S)
NIELIT Chandigarh

sarwan@nielit

The whole purpose of education is to turn mirrors into windows.
- Harris



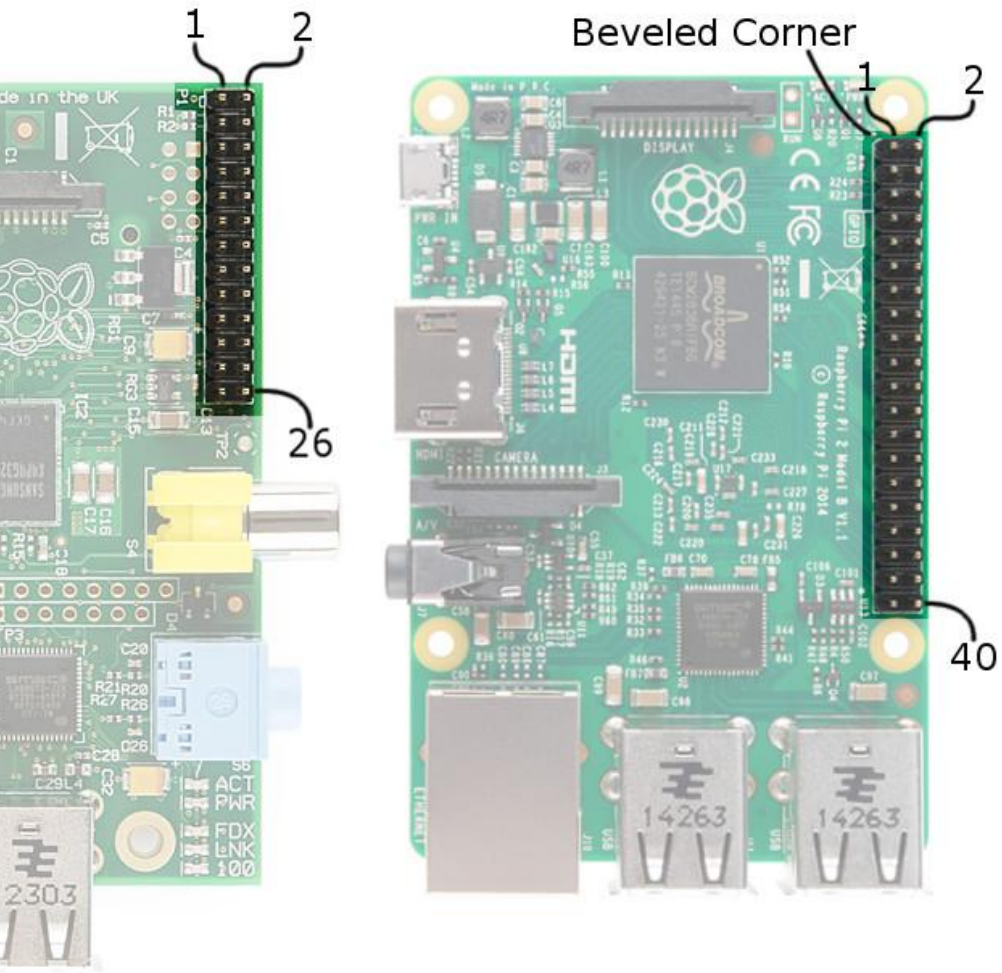
Raspberrry Pi



- Irrespective of its size, Raspberry Pi is a powerhorse of a computer. It can drive HDMI displays, mouse, keyboard, camera – above all it runs full featured Linux distribution.
- Not only computer it is hardware prototyping tool.
- The Pi has **bi-directional I/O pins**, which can be used to drive LEDs, spin motors, or read button presses.

sarwan@nielit

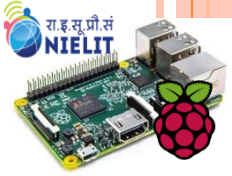
GPIO Pinout



When referencing Pi pin numbers, there are two different numbering schemes:

- Broadcom chip-specific pin numbers (BCM)
- P1 physical pin numbers.

sarwan@nielit



Raspberry Pi2 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Early Models

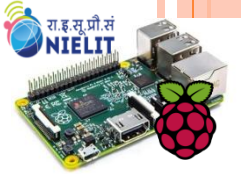
Late Models

Rev. 1
26/01/2014

http://www.element14.com

sarwan@nielit

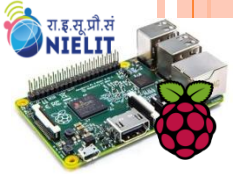
GIT & GITHUB



- The purpose of **Git** is to manage a project, or a set of files, as they change over time.
- **Git** stores this information in a data structure called a **repository**.
- A **git repository** contains, among other things-A set of commit objects. A set of references to commit objects, called heads.
- **GitHub** is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, **GitHub** provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

sarwan@nielit

C (WiringPi) Setup



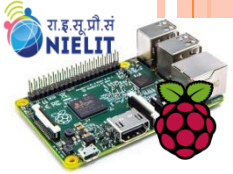
- Install Wiring Pi

  WiringPi is not included with Raspbian, steps to download

  $ git clone git://git.drogon.net/wiringPi
  $ cd wiringPi
  $ git pull origin
  $ cd wiringPi
  $ ./build

sarwan@nielit

Test Wiring Pi



- WiringPi is a C library, it includes a **command-line utility** as well. It can be tested from command line:

  $ gpio -g mode 18 output
  $ gpio -g write 18 1
  $ gpio -g write 18 0

if LED is connected to pin 18 it should blink on and off following the last two commands.

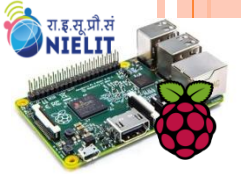
- To test the button, type:

  $ gpio -g mode 17 up
  $ gpio -g read 17

  Either 0 or 1 will be returned, depending on whether the button is pressed or not.

sarwan@nielit

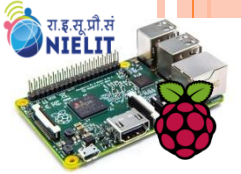
C (WiringPi) Program



```
#include <wiringPi.h>
wiringPiSetup(); // Initializes wiringPi using wiringPi's simlified number system.
wiringPiSetupGpio(); // Initializes wiringPi using the Broadcom GPIO pin numbers
//pin mode selection
pinMode(17, INPUT);
pinMode(23, OUTPUT);
pinMode(18, PWM_OUTPUT);
```

sarwan@nielit

C (WiringPi) Program …



Digital Output

The digitalWrite([pin], [HIGH/LOW]) function

digitalWrite(23, HIGH); //To set pin 23 as HIGH

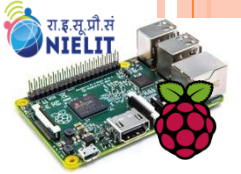
PWM ("Analog") Output

For PWM function is pwmWrite([pin], [0-1023])

pwmWrite(18, 723);

sarwan@nielit

C (WiringPi) Program …



Digital Input

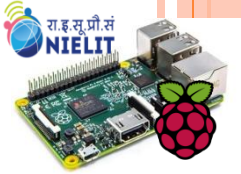
To read the digital state of a pin function is digitalRead([pin])

```
if (digitalRead(17))
   printf("Pin 17 is HIGH\n");
else
   printf("Pin 17 is LOW\n");
```

will print the status of pin 22.

sarwan@nielit

C (WiringPi) Program …



Pull-up/pull-down resistors

pullUpDnControl([pin], [PUD_OFF, PUD_DOWN, PUD_UP]) function is used to pull pin.

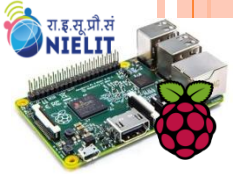
e.g. a button on pin 22 needs to be pulled up

pullUpDnControl(17, PUD_UP);

easy to test if button pulls low when it is pressed.

sarwan@nielit

C (WiringPi) Program …



Delay

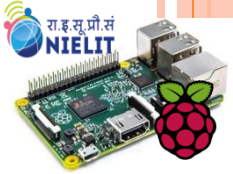
- delay([milliseconds])
- delayMicroseconds([microseconds]).

The standard delay will halt the program flow for a specified number of milliseconds.

delay(2000); // delay for 2 seconds

sarwan@nielit

BLINK

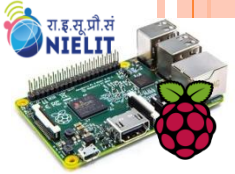


blink.c:

```
#include <wiringPi.h>
int main (void)
{ wiringPiSetup () ;
  pinMode (1, OUTPUT) ;
  for (;;) {
      digitalWrite (1, HIGH) ; delay (500) ;
      digitalWrite (1, LOW) ; delay (500) ;
  } return 0 ;
}
```

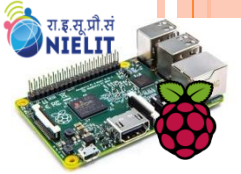
sarwan@nielit

CREATING BLINKER.C



```
$ mkdir c_example
$ cd c_example
$ touch blinker.c
$ leafpad blinker.c &
```

sarwan@nielit



Compile and Execute!

- To **compile our program**, invoke gcc, at terminal

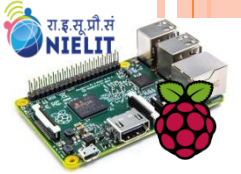
  $ gcc -o blinker blinker.c -l wiringPi

- Type this to **execute your program**:

  $ sudo ./blinker

sarwan@nielit

Download and Install Geany IDE

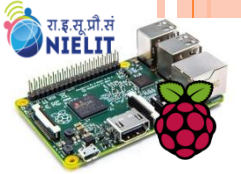


```
$ sudo apt-get update
$ sudo apt-get install geany
```

```
$ sudo geany blinker.c
```

sarwan@nielit

RUNNING BLINK.C



Then to compile and run, enter:

- **gcc -Wall -o blink blink.c -lwiringPi**
- **sudo ./blink**

- *examples* directory of the ***wiringPi*** distribution To use the make file to compile them:

make blink

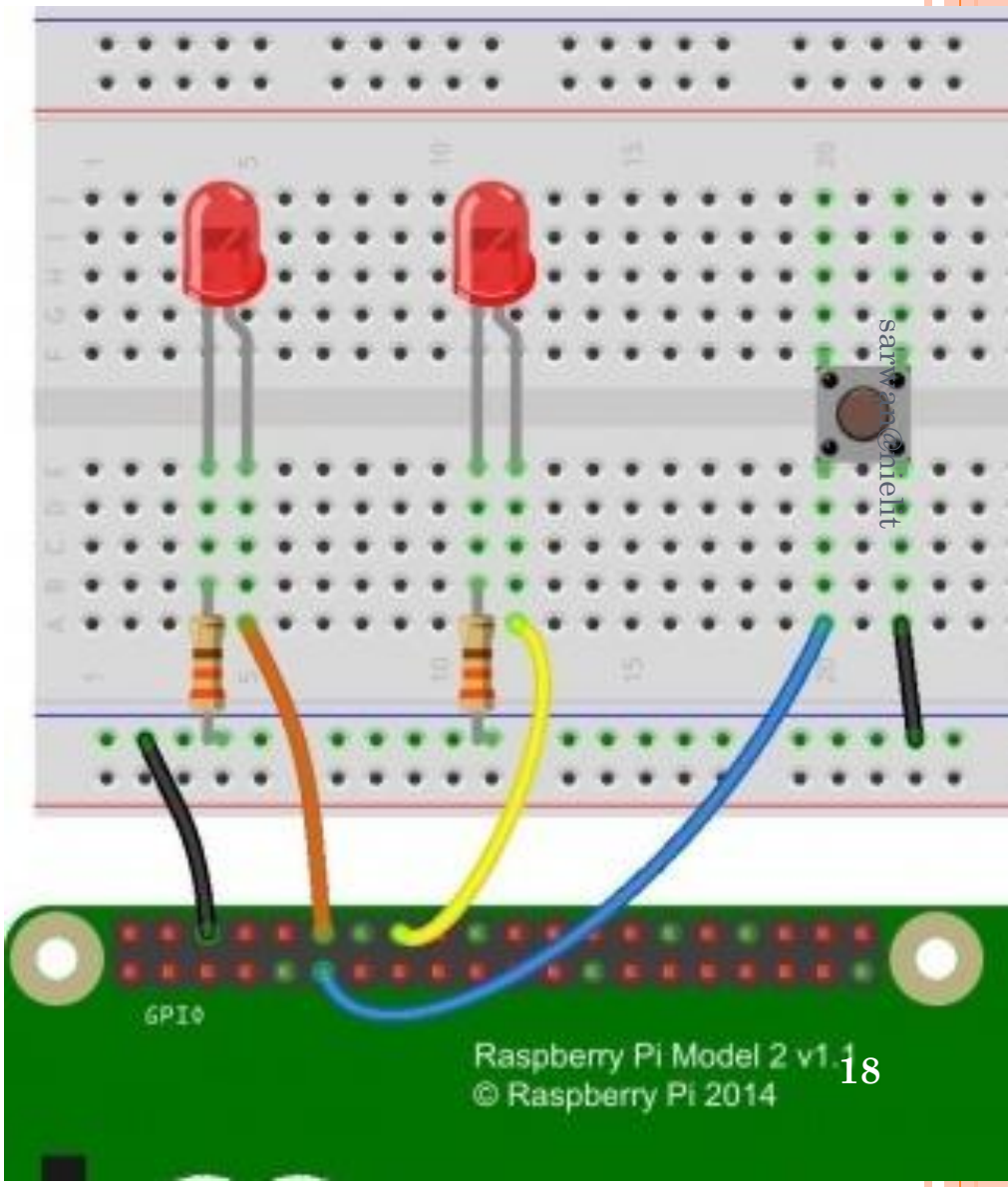
make blink8

make blink12

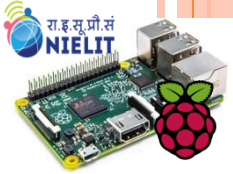
sarwan@nielit

BUTTON INTERFACING

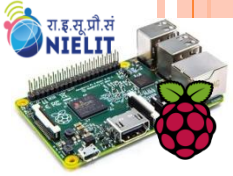
- **two LEDs** are connected to the **Pi's GPIO 18 and GPIO 23,** P1 connector pin numbers, that'd be pins 12 and 16.
- **button** is connected to Broadcom **GPIO 17**, aka P1 pin 11



CODIFY

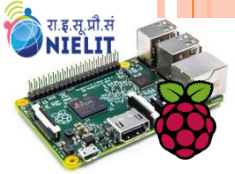


- #include <stdio.h> // Used for printf() statements
- #include <wiringPi.h> // Include WiringPi library!
- // Pin number declarations. We're using the Broadcom chip pin numbers.
- const int pwmPin = 18; // PWM LED - Broadcom pin 18, P1 pin 12
- const int ledPin = 23; // Regular LED - Broadcom pin 23, P1 pin 16
- const int butPin = 17; // Active-low button - Broadcom pin 17, P1 pin 11
- const int pwmValue = 75; // Use this to set an LED brightness



```
int main(void)
{
    // Setup stuff:
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers
    pinMode(pwmPin, PWM_OUTPUT);
        // Set PWM LED as PWM output
    pinMode(ledPin, OUTPUT);     // Set regular LED as output
    pinMode(butPin, INPUT);      // Set button as INPUT
    pullUpDnControl(butPin, PUD_UP);
        // Enable pull-up resistor on button
    printf("Blinker is running! Press CTRL+C to quit.\n");
```

sarwan@nielit



```
// Loop (while(1)):
    while(1)
    {   if (digitalRead(butPin)) // Button is released if this returns 1
        {    pwmWrite(pwmPin, pwmValue); // PWM LED at bright setting
             digitalWrite(ledPin, LOW);     // Regular LED off
        }else // If digitalRead returns 0, button is pressed
        {    pwmWrite(pwmPin, 1024 - pwmValue);
                  // PWM LED at dim setting
                 // Do some blinking on the ledPin:
             digitalWrite(ledPin, HIGH); // Turn LED ON
             delay(75); // Wait 75ms
             digitalWrite(ledPin, LOW); // Turn LED OFF
             delay(75); // Wait 75ms again
        }
    }   return 0;
}
```

sarwan@nielit