

A decorative graphic on the left side of the slide, consisting of a network of white lines and small circles on a blue gradient background, resembling a circuit board or a neural network.

INTRO TO RPI (PART 3)

BY SUTD IEEE

AGENDA

- RPi Serial Communication (Contd.)
- Using Firebase with RPi (the right way)
- Running a Web Server on the RPi

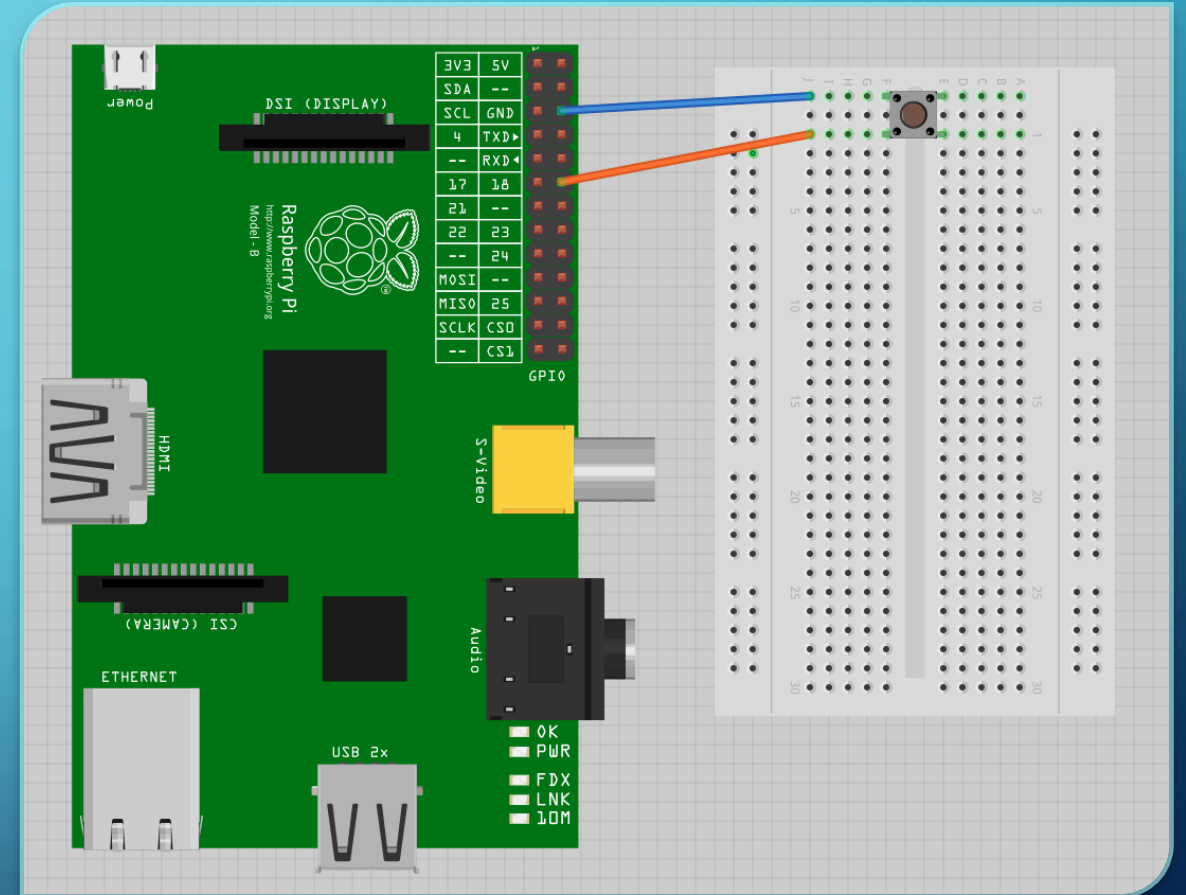
RPI GPIO

<https://pinout.xyz/#>

Pi Model B/B+			
3V3 Power	1	2	5V Power
GPIO2 SDA1 I2C	3	4	5V Power
GPIO3 SCL1 I2C	5	6	Ground
GPIO4	7	8	GPIO14 UART0_TXD
Ground	9	10	GPIO15 UART0_RXD
GPIO17	11	12	GPIO18 PCM_CLK
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3 Power	17	18	GPIO24
GPIO10 SPI0_MOSI	19	20	Ground
GPIO9 SPI0_MISO	21	22	GPIO25
GPIO11 SPI0_SCLK	23	24	GPIO8 SPI0_CE0_N
Ground	25	26	GPIO7 SPI0_CE1_N
ID_SD I2C ID EEPROM	27	28	ID_SC I2C ID EEPROM
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12
GPIO13	33	34	Ground
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
Ground	39	40	GPIO21
Pi Model B+			

ACTIVITY: PUSH BUTTON

- Connect one end of the button to BCM26
- Connect the other end on the same side to GND



ACTIVITY: PUSH BUTTON

- `GPIO.setup(26, GPIO.IN, pull_up_down=GPIO.PUD_UP)`
- `GPIO.input(26)`

ACTIVITY: PUSH BUTTON - DEBOUNCE

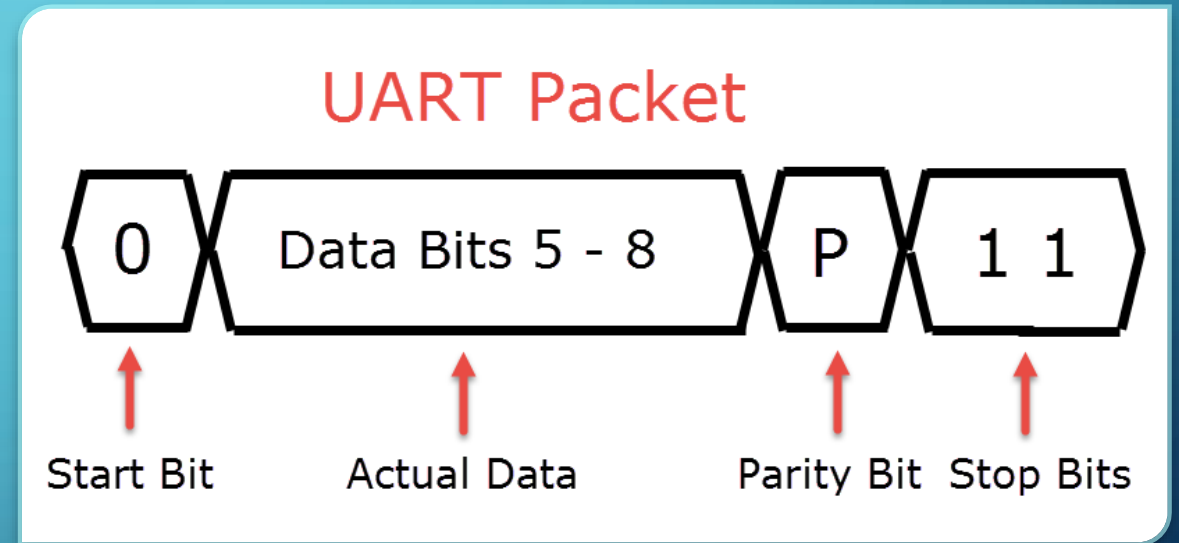
- Oscillation in mechanical switch in button => Multiple button presses
- Logic:
 - Wait for x ms after button pressed and until button is released
 - Only then register it as 1 button press

SERIAL COMMUNICATION

- Send data bit by bit, instead of all at once
- Many protocols:
 - UART / USART
 - SPI
 - I2C
 - ...

SERIAL COMMUNICATION (UART)

- Universal Asynchronous Receiver-Transmitter
- Start-Bit
- Data Bits
- Parity Bit (Optional)
- Stop Bit/s



PYTHON SERIAL IN RPI (WITH ARDUINO)

- Why?
 - Offload processing and simple tasks to Arduino
 - Add more input/ output pins
 - Connect to other serial peripherals
- How?
 - Connect a USB cable from the Pi to the Arduino

PYTHON SERIAL IN RPI (WITH ARDUINO)

- Install pyserial

- `sudo pip3 install pyserial`

- Create a Python file

- `import serial`

- `ser = serial.Serial('/dev/tty*', 9600)`

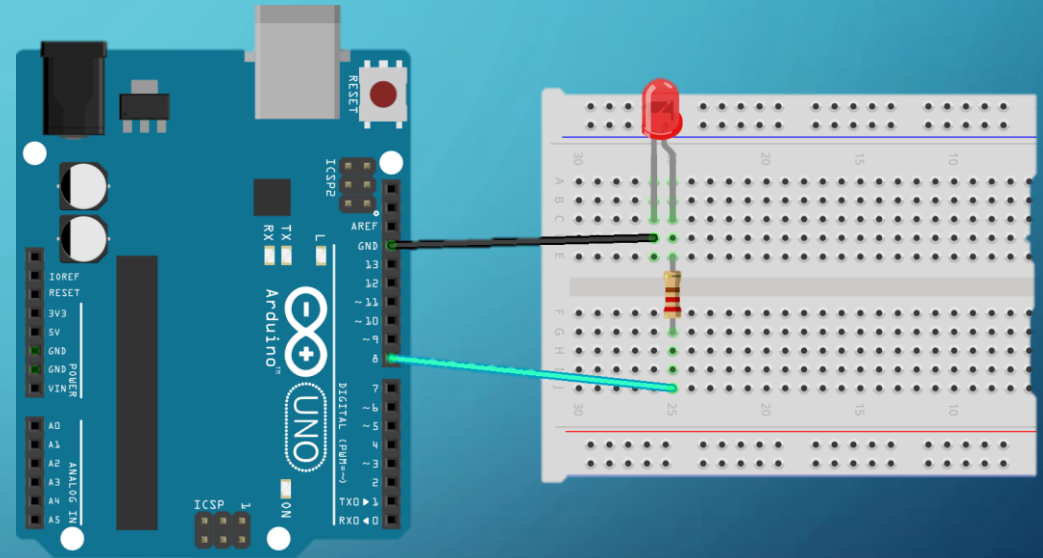
- `ser.write('Something')` >> 'Something' must be in *bytes*

- Check serial port name:

- `ls /dev/tty*`

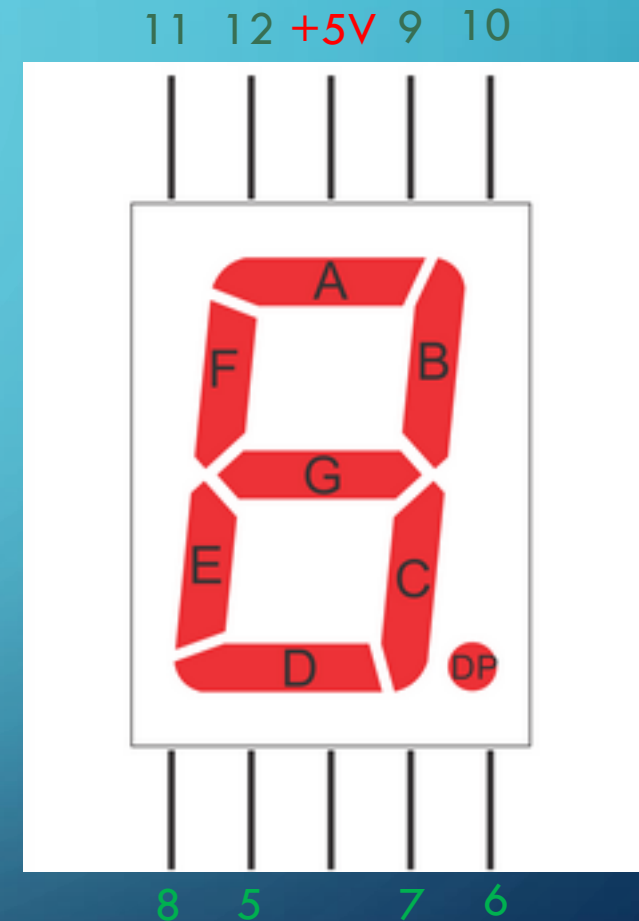
ON ARDUINO

```
• String inString = "";  
void serialEvent() {  
    char c = ' ';  
    c = Serial.read();  
    if (c != '\n') {  
        inString += c;  
    } else {  
        // Do Something !  
        inString = "";  
    }  
}
```



ON ARDUINO

- Seven Segment Display
 - Common Anode
 - Connect one of the 'Com' to +5V
 - Connect other pins to Arduino Digital Pins
 - Through RESISTORS!



SPI AND I2C

- RPi has:
 - 3 SPI Bus's (Only one accessible via the headers)
 - 2 I2C Bus's accessible through the headers
 - I think

FIREBASE (THE RIGHT WAY)

- 'Database Secrets' is deprecated
- 'python-firebase' module not updated in 4 years
- The right way?
 - Firebase Admin SDK (Python)

FIREBASE (THE RIGHT WAY)

- Open your Firebase Console
- Navigate to Project Settings >> Service Accounts
- Generate new Private Key >> Save that file in your project folder

FIREBASE (THE RIGHT WAY)

- In the Firebase Console:
- Open Database Tab
- Navigate to 'Rules'

- Change it to this:

```
{  
  "rules": {  
    ".read": "auth.uid === `some_UID`",  
    ".write": "auth.uid === `some_UID`"  
  }  
}
```

- Publish

FIREBASE (THE RIGHT WAY)

- How to install?

- `sudo pip3 install firebase-admin`

- Create a Python file

- `import firebase_admin`
 - `from firebase_admin import credentials`
 - `from firebase_admin import db`

FIREBASE (THE RIGHT WAY)

```
cred = credentials.Certificate('mykey.json')
firebase_admin.initialize_app(cred, {
    'databaseURL':
    'https://your_database.firebaseio.com/',
    'databaseAuthVariableOverride': {
        'uid': 'your_uid'
    }
})
```

FIREBASE (THE RIGHT WAY)

- To use the database:

- `mydatabase = db.reference()`

- **Get:**

- `mydatabase.get()`

- `mydatabase.child('some_child').get()`

- **Set:**

- `mydatabase.child('some_child').set({ "key": "value" })`

- **Update**

- **Push**

FIREBASE (THE RIGHT WAY)

- Refer to the template to learn how to continuously get a certain database entry

FIREBASE (THE RIGHT WAY)

- Your task:
 - Write another file that asks for an `input` and updates a particular database entry using `.set()` -> On your computer
 - Run a script on the RPi to continuously read that same database entry and based on that asks the Arduino to display said input on the 7-Segment Display

FLASK SERVER

- Flask is a Python framework that allows you to manage your web servers
 - It is not a web server! (Though it comes with one by default)

- Install:

- virtualenv:

- `sudo pip3 install virtualenv`
 - `mkdir myproject`
 - `cd myproject`
 - `virtualenv myvenv`
 - `. myvenv/bin/activate`

- flask:

- `pip3 install Flask`

FLASK SERVER

- Refer to 'flaskapp' in Templates for example

The background is a blue gradient with abstract white lines resembling circuit traces or data paths in the corners. These lines connect small circles, some of which are larger than others, creating a network-like structure. The lines are more prominent in the top-left and bottom-left corners, and less so in the top-right and bottom-right corners.

THE END