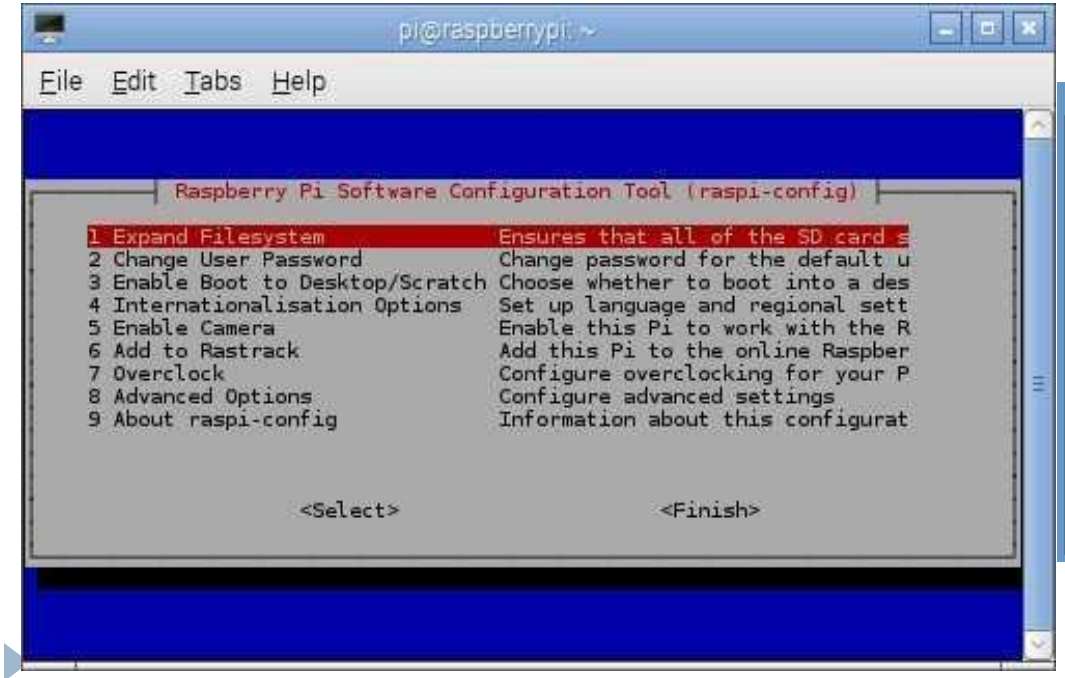
IOT & Applications

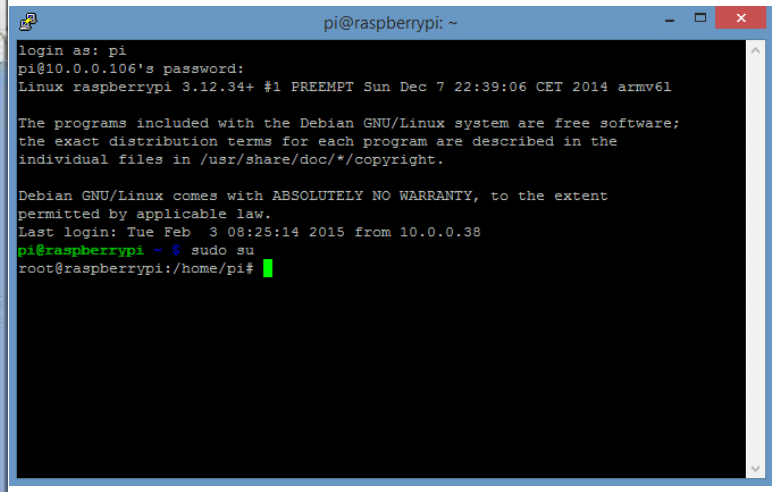
Module - 4

IoT using Arduino: Raspberry Pi Programming, SPI, I2C

Some important commands

- To open the configuration tool after this, simply run the following from the command line:
  - sudo raspi-config
- The sudo is required because you will be changing files that you do not own as the pi user.





GENERAL COMMANDS

•apt-get update: Synchronizes the list of packages on your system to the list in the repositories.
Use it before installing new packages to make sure you are installing the latest version.
•apt-get upgrade: Upgrades all of the software packages you have installed.
•clear: Clears previously run commands and text from the terminal screen.
•date: Prints the current date.
•find / -name example.txt: Searches the whole system for the file example.txt and outputs a list of all directories that contain the file.
•nano example.txt: Opens the file example.txt in the Linux text editor Nano.
•poweroff: To shutdown immediately.
•raspi-config: Opens the configuration settings menu.
•reboot: To reboot immediately.
•shutdown -h now: To shutdown immediately.
•shutdown -h 01:22: To shutdown at 1:22 AM.
•startx: Opens the GUI (Graphical User Interface).

FILE AND DIRECTORY COMMANDS

•cat example.txt: Displays the contents of the file example.txt.
•cd /abc/xyz: Changes the current directory to the /abc/xyz directory.
•cp XXX: Copies the file or directory XXX and pastes it to a specified location; i.e. cp examplefile.txt /home/pi/office/ copies examplefile.txt
in the current directory and pastes it into the /home/pi/ directory. If the file is not in the current directory, add the path of the file's location
(i.e. cp /home/pi/documents/examplefile.txt /home/pi/office/ copies the file from the documents directory to the office directory).
•ls -l: Lists files in the current directory, along with file size, date modified, and permissions.
•mkdir example_directory: Creates a new directory named example_directory inside the current directory.
•mv XXX: Moves the file or directory named XXX to a specified location. For example, mv examplefile.txt /home/pi/office/ moves
examplefile.txt in the current directory to the /home/pi/office directory. If the file is not in the current directory, add the path of the
file's location (i.e. cp /home/pi/documents/examplefile.txt /home/pi/office/ moves the file from the documents directory to the office directory).
This command can also be used to rename files (but only within the same directory). For example, mv examplefile.txt newfile.txt renames
examplefile.txt to newfile.txt, and keeps it in the same directory.
•rm example.txt: Deletes the file example.txt.
•rmdir example_directory: Deletes the directory example_directory (only if it is empty).

SYSTEM INFORMATION COMMANDS

•cat /proc/meminfo: Shows details about your memory.
•cat /proc/partitions: Shows the size and number of partitions on your SD card or hard drive.
•cat /proc/version: Shows you which version of the Raspberry Pi you are using.
•df -h: Shows information about the available disk space.
•df /: Shows how much free disk space is available.
•dpkg - -get-selections | grep XXX: Shows all of the installed packages that are related to XXX.
•dpkg - -get-selections: Shows all of your installed packages.
•free: Shows how much free memory is available.
•hostname -I: Shows the IP address of your Raspberry Pi.
•lsusb: Lists USB hardware connected to your Raspberry Pi.
•UP key: Pressing the UP key will print the last command entered into the command prompt.
This is a quick way to repeat previous commands or make corrections to commands.
•vcgencmd measure_temp: Shows the temperature of the CPU.
•vcgencmd get_mem arm && vcgencmd get_mem gpu: Shows the memory split between the CPU and GPU.

What is SPI protocol

- The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. The interface was developed by Motorola in the mid-1980s and has become a de facto standard. Typical applications include Secure Digital cards and liquid crystal displays.
- SPI devices communicate in full duplex mode using a master-slave architecture with a single master.
- The master device originates the frame for reading and writing. Multiple slave-devices are supported through selection with individual slave select (SS), sometimes called chip select (CS), lines.

SPI Pins and Modes

- SCLK - Serial CLocK
- CE - Chip Enable (often called Chip Select)
- MOSI - Master Out Slave In
- MISO - Master In Slave Out
- MOMI - Master Out Master In

Standard mode

- In Standard SPI mode, the peripheral implements the standard 3 wire serial protocol (SCLK, MOSI and MISO).

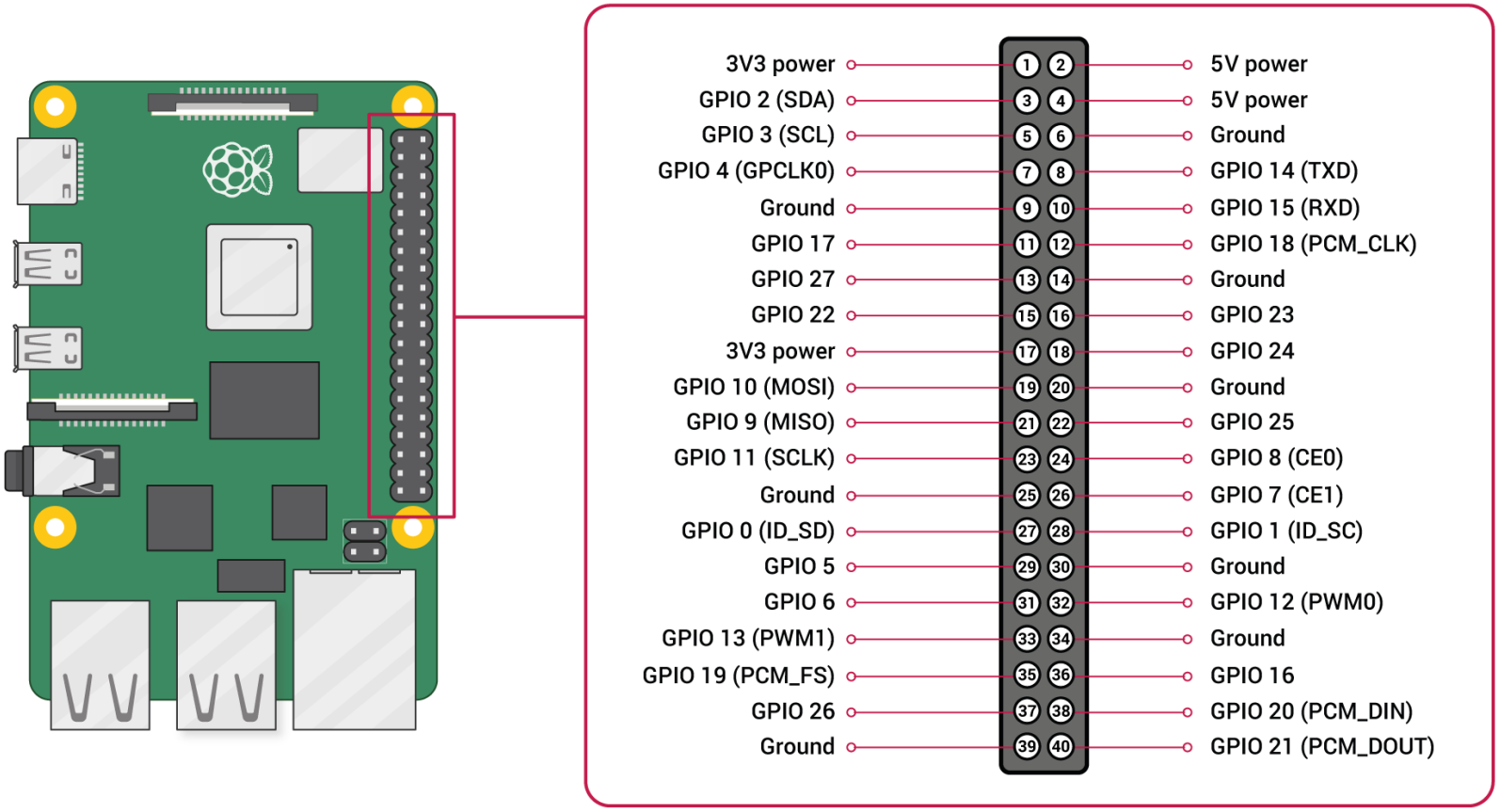
Bidirectional mode

- In bidirectional SPI mode, the same SPI standard is implemented, except that a single wire is used for data (MOMI) instead of the two used in standard mode (MISO and MOSI). In this mode, the MOSI pin serves as MOMI pin.

- MOSI on a master connects to MOSI on a slave. MISO on a master connects to MISO on a slave. Slave Select has the same functionality as chip select and is used instead of an addressing concept.

- Note: on a slave-only device, MOSI may be labeled as SDI (Slave Data In) and MISO may be labeled as SDO (Slave Data Out)

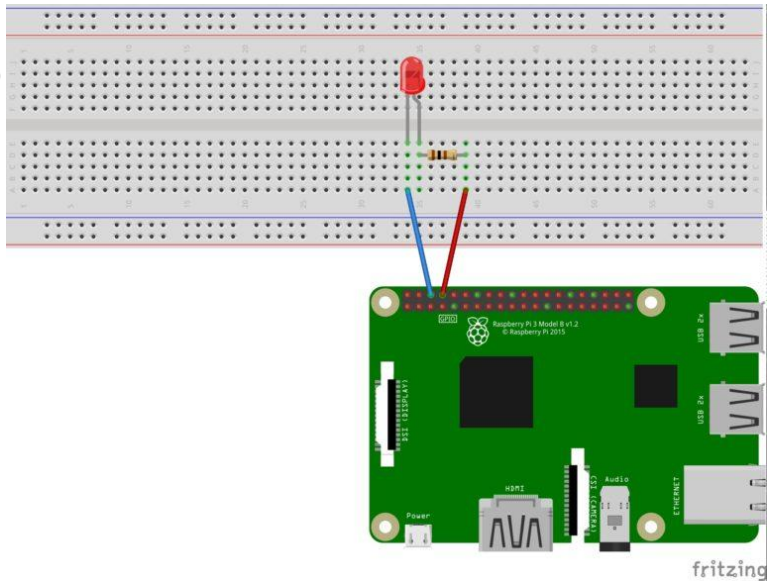
Raspberry Pi Pins



Sample Code-1: LED Blink

```
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
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output
pin and set initial value to low (off)

while True:# Run forever
 GPIO.output(8, GPIO.HIGH) # Turn on
 sleep(1) # Sleep for 1 second
 GPIO.output(8, GPIO.LOW) # Turn off
 sleep(1) # Sleep for 1 second
```

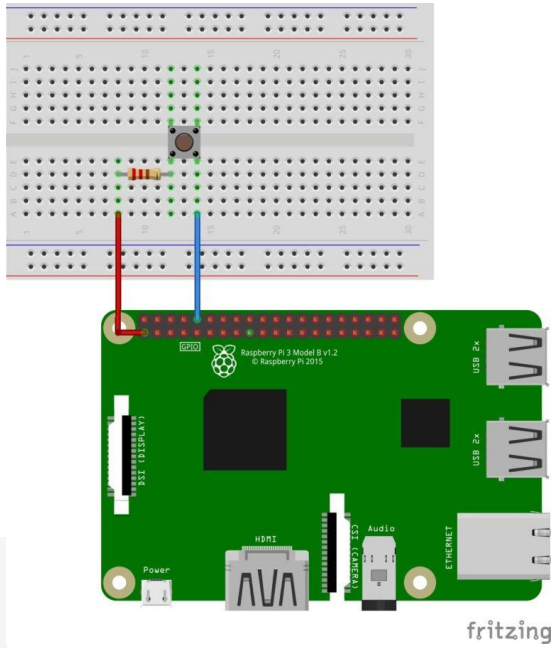
Pin 8: with LED



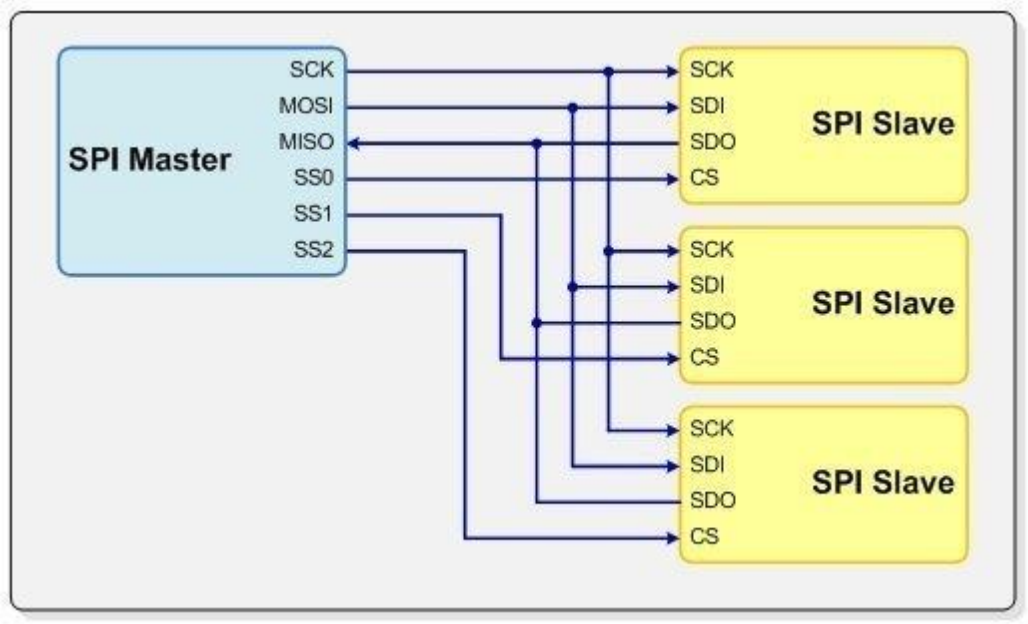
Sample Code-2: Button Event

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library

def button_callback(channel):
    print("Button was pushed!")

GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Set pin 10
to be an input pin and set initial value to be pulled low (off)

GPIO.add_event_detect(10,GPIO.RISING,callback=button_callback) # Setup
event on pin 10 rising edge

message = input("Press enter to quit\n\n") # Run until someone presses
enter

GPIO.cleanup() # Clean up
```

Running the code: $ python3 push_button.py



SPI: **Serial Peripheral Interface**

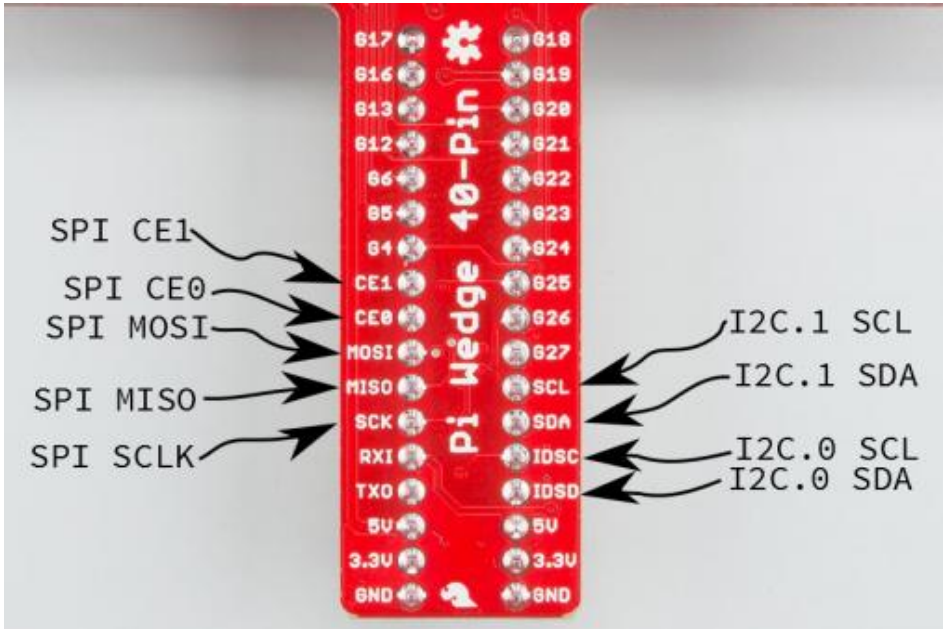


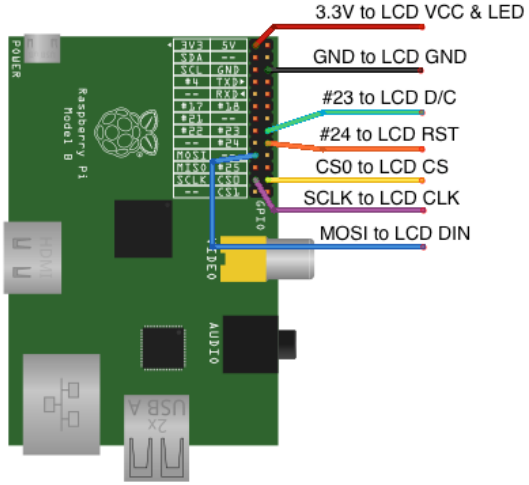
SPI (Serial Peripheral Interface) is an interface bus commonly used for communication with flash memory, sensors, real-time clocks (RTCs), analog-to-digital converters, and more. The Serial Peripheral Interface (SPI) bus was developed by Motorola to provide full-duplex synchronous serial communication between master and slave devices.

- A standard SPI connection involves a master connected to slaves using the serial clock (SCK), Master Out Slave In (MOSI), Master In Slave Out (MISO), and Slave Select (SS) lines. The SCK, MOSI, and MISO signals can be shared by slaves while each slave has a unique SS line.

Pin Connection: SPI LCD with RPi

Raspberry Pi Signal	Serial 7-seg Signal
GND	GND
3.3V	VCC
CE1	SS (Shift Select)
SCK	SCK
MOSI	SDI
MISO	SDO





Sample Code-3: SPI LCD with RPi

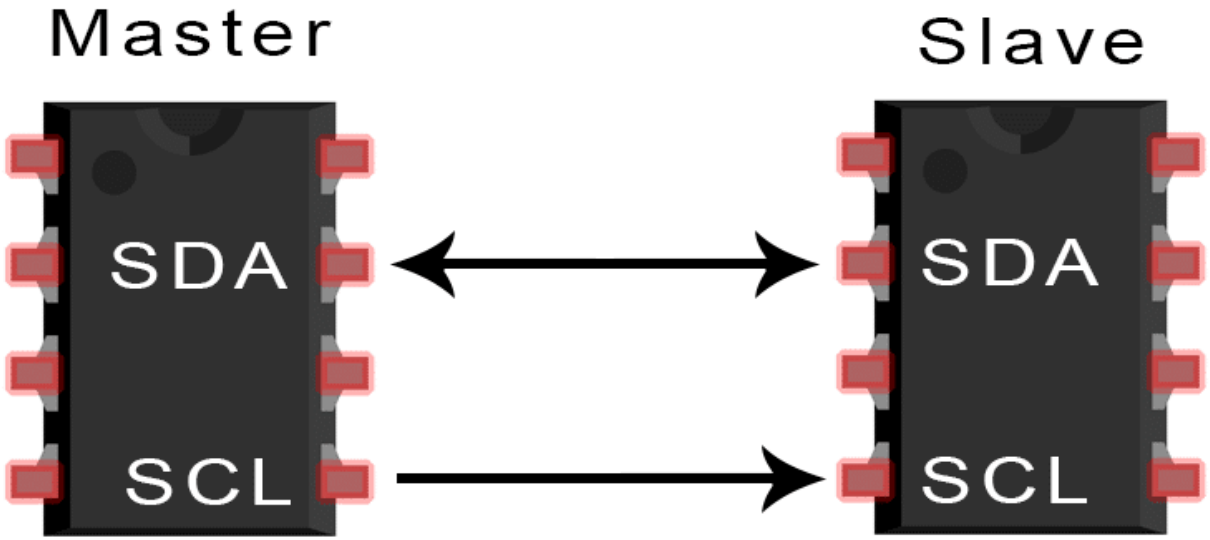
```
# Clear display
msg = [0x76]
spi.xfer2(msg)

time.sleep(5)

# Turn on one segment of each character to show that we can
# address all of the segments
i = 1
while i < 0x7f:

    # The decimals, colon and apostrophe dots
    msg = [0x77]
    msg.append(i)
    result = spi.xfer2(msg)

    # The first character
    msg = [0x7b]
    msg.append(i)
    result = spi.xfer2(msg)

    # The second character
    msg = [0x7c]
    msg.append(i)
    result = spi.xfer2(msg)

    # The third character
    msg = [0x7d]
    msg.append(i)
    result = spi.xfer2(msg)

    # The last character
    msg = [0x7e]
    msg.append(i)
    result = spi.xfer2(msg)

    # Increment to next segment in each character
    i <<= 1

    # Pause so we can see them
    time.sleep(5)

# Clear display again
msg = [0x76]
spi.xfer2(msg)
```

I2C Protocol

- I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line). Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave.
- I2C combines the best features of SPI and UARTs. With I2C, multiple slaves to a single master (like SPI) can be connected and multiple masters controlling single, or multiple slaves is also possible. This is really useful when you want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD.
- Like UART communication, I2C only uses two wires to transmit data between devices:

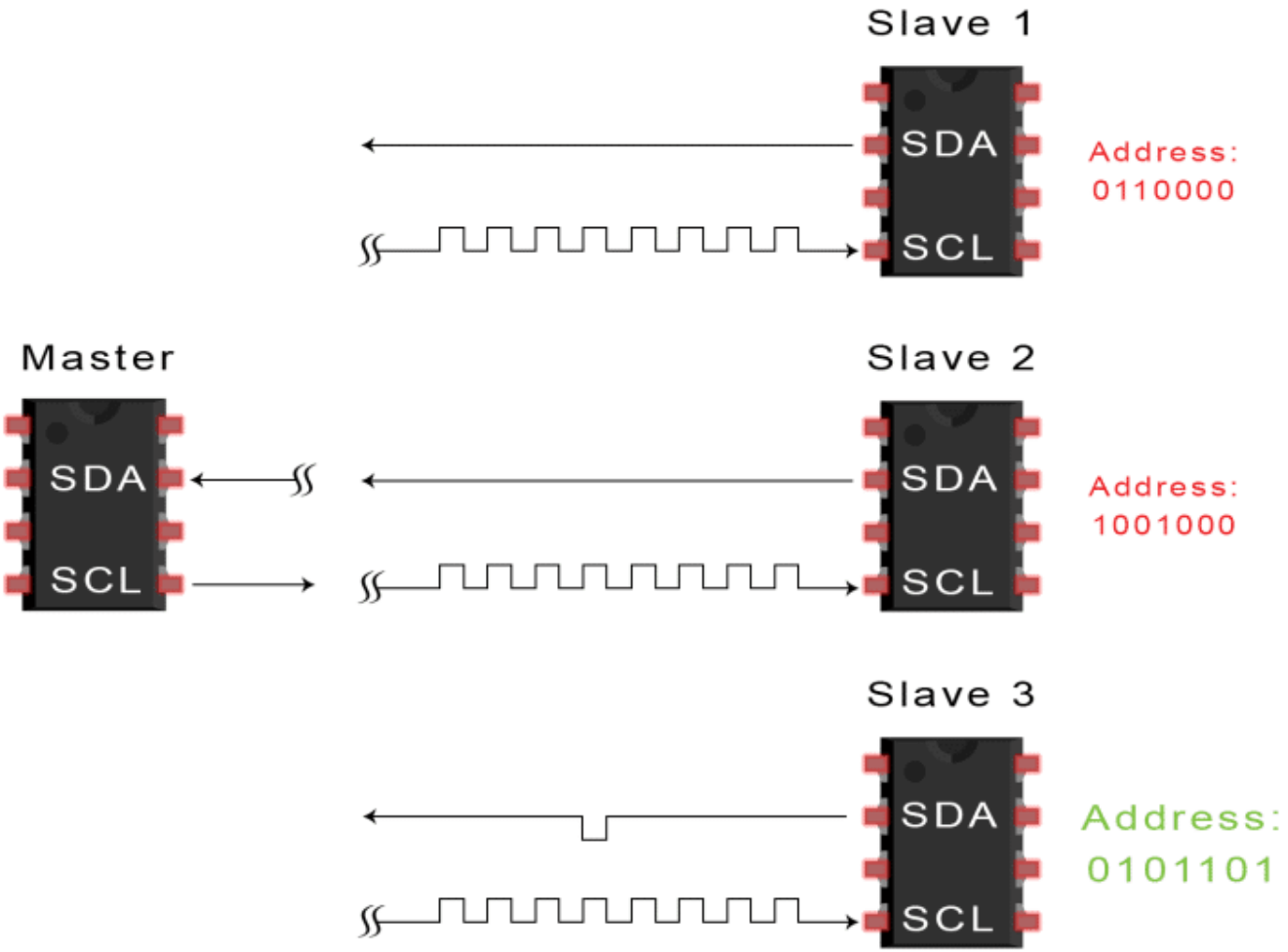
- The I2C communication bus is very popular and broadly used by many electronic devices because it can be easily implemented in many electronic designs which require communication between a master and multiple slave devices or even multiple master devices.
- The easy implementations comes with the fact that only two wires are required for communication between up to almost 128 devices when using 7 bits addressing and up to almost 1024 devices when using 10 bits addressing.

I2C Connection

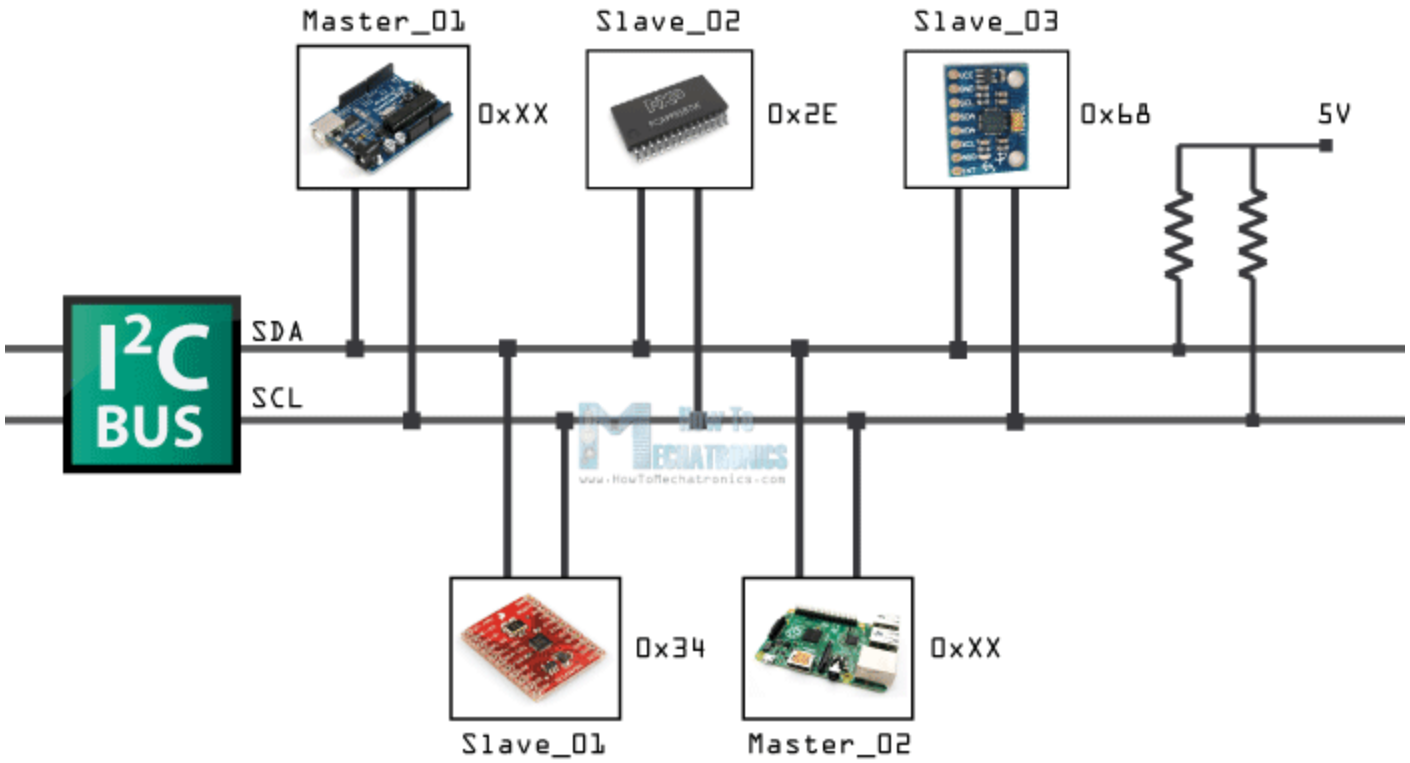


SDA (Serial Data) – The line for the master and slave to send and receive data.
SCL (Serial Clock) – The line that carries the clock signal.

I2C Multi Device Connection



I2C Bus architecture



THANK YOU