GPIO Output (LED Blink)

```
import RPi.GPIO as GPIO
import time

LED_PIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

while True:
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(1)
```

GPIO Input (Button Press)

```
import RPi.GPIO as GPIO

BUTTON_PIN = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    if GPIO.input(BUTTON_PIN) == GPIO.LOW:
        print("Button Pressed")
```

PWM Signal Generation

```
import RPi.GPIO as GPIO
import time

PWM_PIN = 12
GPIO.setmode(GPIO.BCM)
GPIO.setup(PWM_PIN, GPIO.OUT)

pwm = GPIO.PWM(PWM_PIN, 1000)
pwm.start(50)

time.sleep(5)
pwm.stop()
GPIO.cleanup()
```

SPI Communication (spidev)

```
import spidev

spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz = 50000
```

```
to_send = [0x01, 0x02]
received = spi.xfer(to_send)

print(received)
spi.close()
```

UART Communication (serial)

```
import serial

ser = serial.Serial('/dev/serial0', 9600)
ser.write(b'Hello from Raspberry Pi\n')

while True:
    if ser.in_waiting:
        print(ser.readline().decode('utf-8'))
```

I2C Communication (smbus)

```
import smbus

bus = smbus.SMBus(1)
address = 0x48

bus.write_byte(address, 0x01)
value = bus.read_byte(address)

print("Received:", value)
```

GPIO Interrupt (Edge Detection)

```
import RPi.GPIO as GPIO

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

BUTTON_PIN = 23
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(BUTTON_PIN,         GPIO.FALLING,         callback=button_callback,
bouncetime=300)

message = input("Press Enter to quit\n")
GPIO.cleanup()
```

ADC with MCP3008

```
import spidev
import time
```

```
# SPI setup
spi = spidev.SpiDev()
spi.open(0, 0)  # Bus 0, Device (CS) 0
spi.max_speed_hz = 1350000

# ADC read function
def read_adc(channel):
    assert 0 <= channel <= 7, "MCP3008 has 8 channels (0-7)"
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    value = ((adc[1] & 3) << 8) + adc[2]
    return value

# Main loop
try:
    while True:
        adc_value = read_adc(0)  # Channel 0
        voltage = (adc_value * 3.3) / 1023
        print(f"ADC Value: {adc_value}, Voltage: {voltage:.2f}V")
        time.sleep(1)

except KeyboardInterrupt:
    spi.close()
    print("Program stopped.")
```

Software Timer Example

```
import threading

def task():
    print("Timer Task Executed")

# Set a timer to execute task after 5 seconds
timer = threading.Timer(5.0, task)
timer.start()
```

Software I2C with smbus (Read from RTC DS1307)

```
import smbus
import time

bus = smbus.SMBus(1)
address = 0x68

def read_time():
    def bcd_to_dec(bcd): return (bcd & 0x0F) + ((bcd >> 4) * 10)
    data = bus.read_i2c_block_data(address, 0x00, 7)
    seconds = bcd_to_dec(data[0])
    minutes = bcd_to_dec(data[1])
    hours = bcd_to_dec(data[2])
    print(f"Time: {hours:02}:{minutes:02}:{seconds:02}")
```

```
while True:
    read_time()
    time.sleep(1)
```

UART with pyserial (Loopback Test)

```
import serial
import time

ser = serial.Serial('/dev/serial0', 9600, timeout=1)
ser.flush()

while True:
    ser.write(b"Hello UART\n")
    time.sleep(1)
    if ser.in_waiting > 0:
        line = ser.readline().decode('utf-8').rstrip()
        print(line)
```