```
#Sara Falkson: PiPico Accelerometer and LED

from machine import Pin, I2C
from time import sleep

# Constants and MPU6050 setup
MPU6050_ADDR = 0x68
PWR_MGMT_1 = 0x6B
TEMP_OUT_H = 0x41
ACCEL_XOUT_H = 0x3B
GYRO_XOUT_H = 0x43

# Initialize I2C for MPU6050
i2c = I2C(0, scl=Pin(9), sda=Pin(8), freq=400000)  # Setup I2C on GPIO 8 and 9

# Initialize external LED and button
led = Pin(15, Pin.OUT)
button = Pin(16, Pin.IN)

# Helper functions for MPU6050
def write_byte(addr, reg, data):
    i2c.writeto_mem(addr, reg, bytearray([data]))

def read_word(addr, reg):
    data = i2c.readfrom_mem(addr, reg, 2)
    value = (data[0] << 8) | data[1]
    return value if value < 0x8000 else value - 0x10000

def init_mpu6050():
    write_byte(MPU6050_ADDR, PWR_MGMT_1, 0)  # Wake up the MPU6050

def read_accel():
    return (read_word(MPU6050_ADDR, ACCEL_XOUT_H),
            read_word(MPU6050_ADDR, ACCEL_XOUT_H + 2),
            read_word(MPU6050_ADDR, ACCEL_XOUT_H + 4))

def read_gyro():
    return (read_word(MPU6050_ADDR, GYRO_XOUT_H),
            read_word(MPU6050_ADDR, GYRO_XOUT_H + 2),
            read_word(MPU6050_ADDR, GYRO_XOUT_H + 4))

def read_temp():
```

```
    temp_raw = read_word(MPU6050_ADDR, TEMP_OUT_H)
    return (temp_raw / 340.0) + 36.53

# Initialize MPU6050
init_mpu6050()

# Main loop
while True:
    # Read sensor data
    accel_x, accel_y, accel_z = read_accel()
    gyro_x, gyro_y, gyro_z = read_gyro()
    temp_celsius = read_temp()
    print(f"Accelerometer X: {accel_x} Y: {accel_y} Z: {accel_z}")
    print(f"Gyroscope X: {gyro_x} Y: {gyro_y} Z: {gyro_z}")
    print(f"Temperature: {temp_celsius:.2f} C")

    # LED control based on button press
    if button.value() == 1:
        led.on()
    else:
        led.off()

    sleep(.01) # Short sleep to reduce CPU usage without losing much responsiveness
```