Tutorial 11: Smart monitoring system

1. Introduction

1.1 What am I learning here and why?

An intelligent, integrated smart monitoring system can support you in your daily life. It allows you to remotely monitor your home installations, i.e. check security camera feeds, receive real-time alerts and notifications, and manage various aspects of your home's security and automation from anywhere.

In this tutorial, you will use the sensors and components that you used in the previous tutorials in order to create a smart monitoring system for your SmartHome house model. You will need to use the knowledge acquired since the first tutorial and put it into practice.

The goal is to utilize the OLED display in order to display information on all connected sensors and electronics.

1.2 Learning Objectives

After you have completed the tutorial you will

- Understand the use and characteristics of the OLED display.
- Be able to create a smart monitoring system that provides information on all connected electronics and sensors.

1.3 What do I need?

Software

So that you can carry out the installations shown in this tutorial you should have downloaded the Thonny programming environment on your device. Also, you need to have installed the firmware of MicroPython on your Raspberry Pi Pico. The extended modifications (see p 42 in manual) including the extra components and their connectivity must also be made on the breadboard.

Electrical Hardware

- 1 x Raspberry Pi Pico
- 1 x Full-size breadboard
- 1 x Micro-USB cable
- Jumper wires as needed
- 1 x OLED SSD1306 display
- 5 x 220 Ohm resistor
- 2 x 1k Ohm resistor
- 1 x LDR photoresistor
- 1 x HC-SR04 ultrasonic sensor

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- 1 x 100nF Capacitor
- 1 x Push button
- 1 x PIR motion detector sensor
- 4 x LEDs
- 1 x DHT11 sensor
- 1 x Traffic lights LED
- 2 x SG90 servo motor
- 1 x MQ-135 air quality sensor
- 1 x Fan
- 1 x Buzzer
- 1 x Flame detection sensor
- 1 x TIP-120 control module
- 1 x Raindrop sensor
- 1 x Diode 1N40007
- 1 x RFID reader RC522
- 1 x Rotary potentiometer
- 1 x RFID key fob

To attach components at SmartHome4Seniors house model

Please check out all previous tutorials and the manuals for technical composition of the house model. Completion of the previous tutorials is considered a precondition for this tutorial.

Ability

As regards physical skills you should be able to count off holes on the breadbord and insert components to it.

2. Learning content

2.1 Theoretical background

What is an OLED SSD1306 display?

An OLED SSD1306 display is a type of electronic display that uses OLED (Organic Light-Emitting Diode) technology and is controlled by an SSD1306 display driver IC (Integrated Circuit). OLED is a display technology that uses organic compounds to emit light when an electric current is applied. Unlike traditional LCD displays, OLEDs do not require a backlight, which allows them to be thinner, lighter, and more power-efficient. OLED displays offer vibrant colors, high contrast ratios, and fast response times, making them suitable for various applications, including smartphones, TVs, and small screens like those found in wearables and IoT devices. The SSD1306 is an OLED display driver IC

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

(Integrated Circuit) manufactured by Solomon Systech. This driver IC is designed to control OLED displays and is commonly used for small monochrome OLED screens. It communicates with a microcontroller or other control circuitry to display text, graphics, and images on the OLED screen. The SSD1306 IC is compatible with various microcontrollers and development platforms, making it a popular choice for hobbyists and developers.

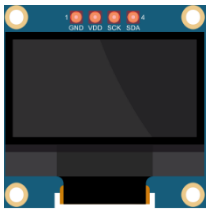


Figure 1 OLED SSD1306 display

OLED SSD1306 displays are often used in DIY electronics projects, embedded systems, and wearable devices because of their compact size, low power consumption, and sharp image quality. These displays are available in various sizes, resolutions, and configurations, and they are commonly used in applications like smartwatches, fitness trackers, digital cameras, and IoT devices where space and power efficiency are important considerations.

Developers can program the SSD1306 display to show text, graphics, and even simple animations, making it a versatile choice for adding a visual interface to electronic projects. Libraries and libraries exist to facilitate interfacing with these displays, making it relatively easy to integrate them into different microcontroller-based projects.

To use the OLED SSD1306 display you will need to install the necessary libraries to your Raspberry Pi Pico. Initial information is provided on page of the SmartHome4Seniors manual. However, additional information is provided below in section 2 of this tutorial.

2.2 Step-by-Step Guide

Now let's move on to the implementation of the before mentioned scenario. To do this, take the SmartHome4Seniors house model or just look at the instructions and recordings.

Three steps are required for installing the smart monitoring system using the OLED Display. These are:

1. Install the SSD1306 package to your Raspberry Pi Pico
2. Connect the OLED Display to your Breadboard
3. Write the Micro Python code

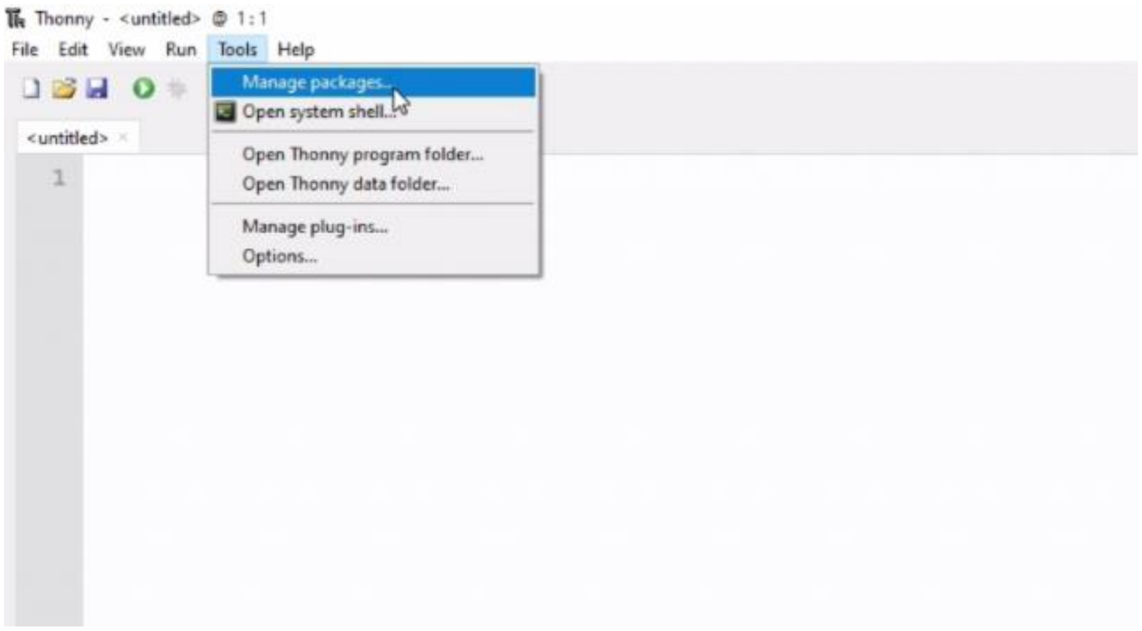
Now, let’s bring our monitoring system to life.

2.2.1 Install the SSD1306 package to your Raspberry Pi Pico

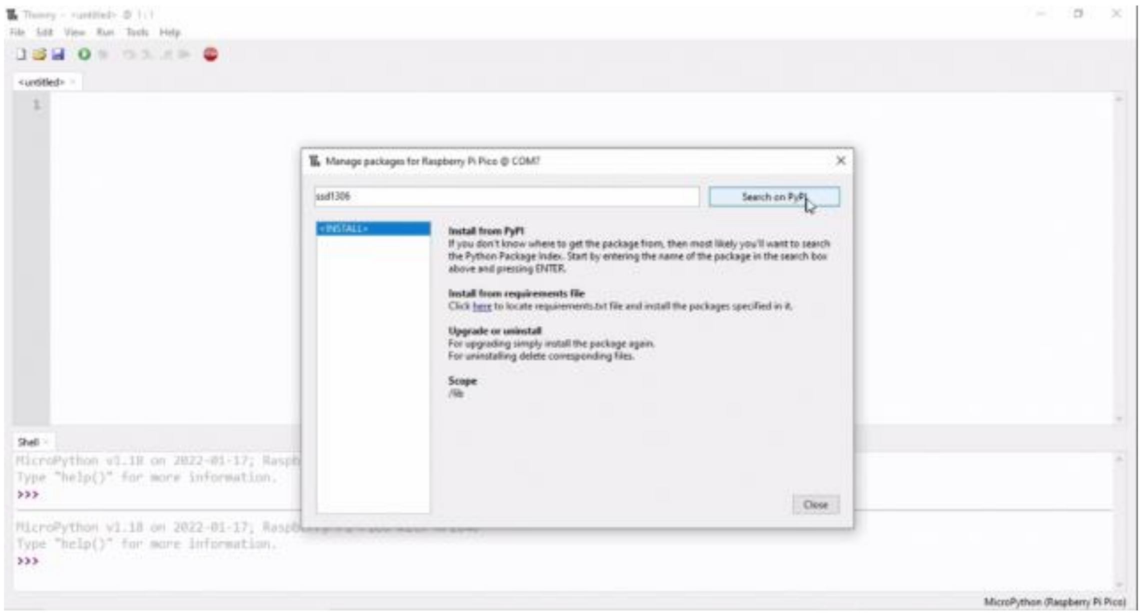
Before starting programming, the OLED display, you first need to add the SSD1306 package to your Raspberry Pi Pico. To do that, please follow the next steps:

1. Open Thonny and go to **Tools → Manage packages…**

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled>
Manage packages...
Open system shell...
Open Thonny program folder...
Open Thonny data folder...
Manage plug-ins...
Options...

2. In the manage packages window, type **SSD1306** and click *Search*.


Manage packages for Raspberry Pi Pico @ COM7
ssd1306
Search on PyPI
<INSTALL>
Close
Shell
MicroPython (Raspberry Pi Pico)

Install from PyPI
If you don't know where to get the package from, then most likely you'll want to search the Python Package Index. Start by entering the name of the package in the search box above and pressing ENTER.

Install from requirements file
Click here to locate requirements.txt file and install the packages specified in it.

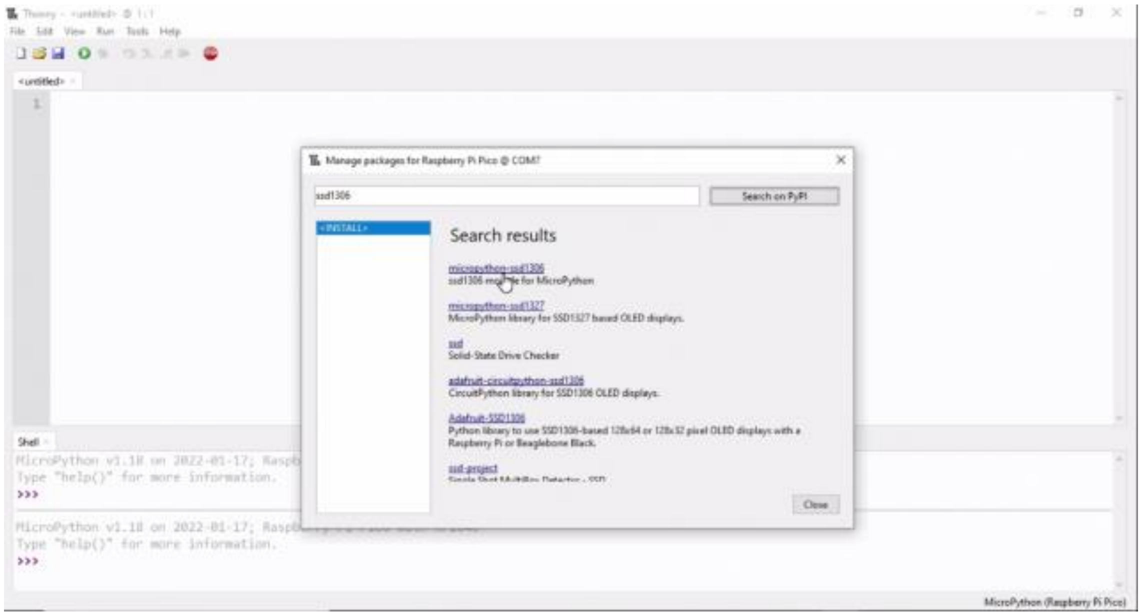
Upgrade or uninstall
For upgrading simply install the package again.
For uninstalling delete corresponding files.

Scope
/lib

```
MicroPython v1.18 on 2022-01-17; Raspb
Type "help()" for more information.
>>>

MicroPython v1.18 on 2022-01-17; Raspb
Type "help()" for more information.
>>>
```

3. Once search is over, click on the *micropython-ssd1306.*


Manage packages for Raspberry Pi Pico @ COM7
ssd1306
Search on PyPI
<INSTALL>
Close
Shell
MicroPython (Raspberry Pi Pico)

Search results

micropython-ssd1306
ssd1306 module for MicroPython

micropython-ssd1327
MicroPython library for SSD1327 based OLED displays.

ssd
Solid-State Drive Checker

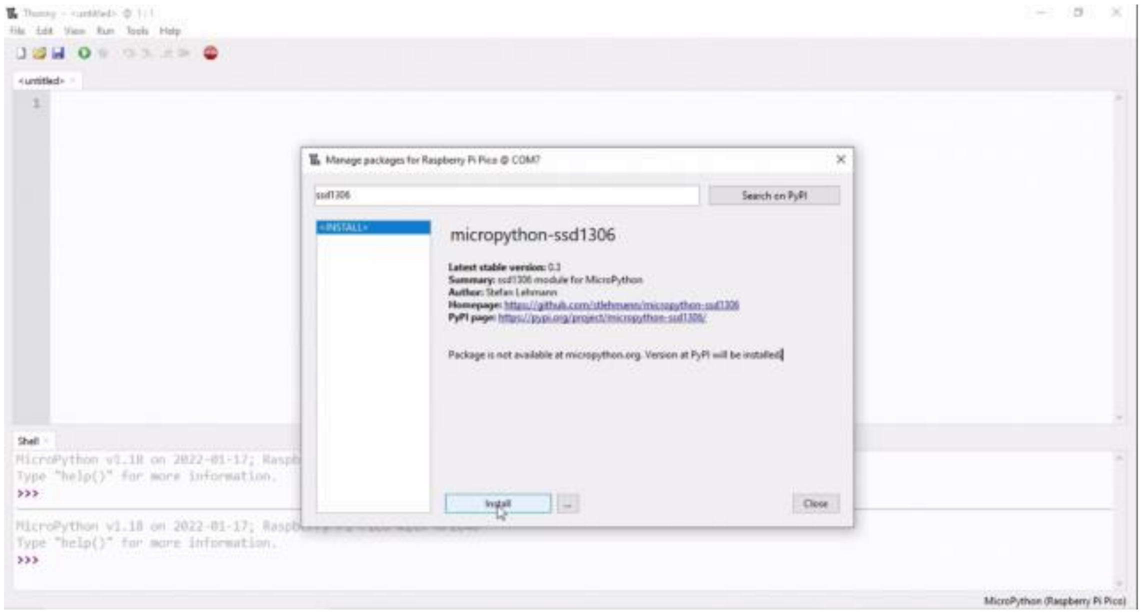
adafruit-circuitpython-ssd1306
CircuitPython library for SSD1306 OLED displays.

Adafruit-SSD1306
Python library to use SSD1306-based 128x64 or 128x32 pixel OLED displays with a Raspberry Pi or Beaglebone Black.

ssd-project

```
MicroPython v1.18 on 2022-01-17; Raspb
Type "help()" for more information.
>>>
```

4. On the next window, click *Install.*


Manage packages for Raspberry Pi Pico @ COM7
ssd1306
Search on PyPI
<INSTALL>
Install
...
Close
Shell
MicroPython (Raspberry Pi Pico)

micropython-ssd1306

Latest stable version: 0.3
Summary: ssd1306 module for MicroPython
Author: Stefan Lehmann
Homepage: https://github.com/stlehmann/micropython-ssd1306
PyPI page: https://pypi.org/project/micropython-ssd1306/

Package is not available at micropython.org. Version at PyPI will be installed.

```
MicroPython v1.18 on 2022-01-17; Raspb
Type "help()" for more information.
>>>
```

5. Wait for package installation, then click close.

In addition, you can always use the ***ssd1306.py*** code which you can find in the appendix of this tutorial.

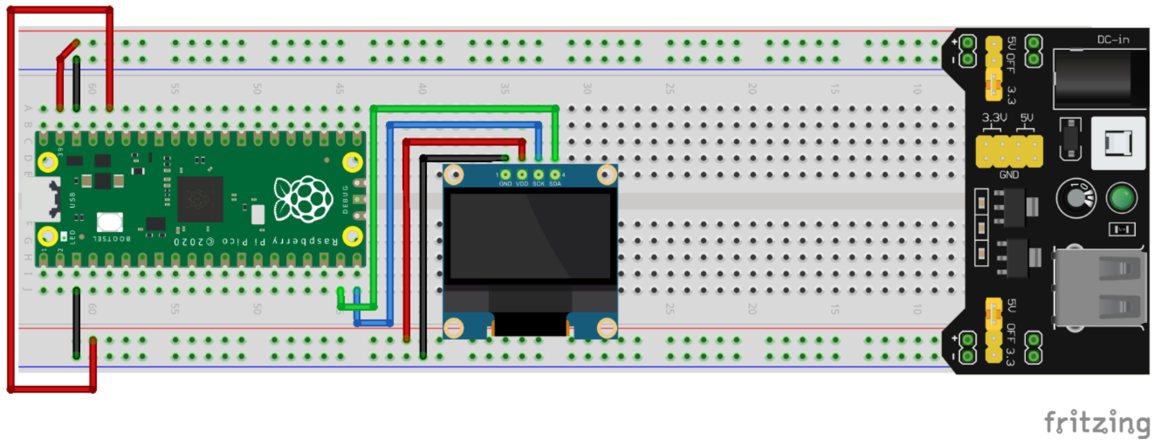
Now, you are ready to proceed with programming the OLED display and all other sensors and electronics.

2.2.2 Connect the OLED Display to your Breadboard

Open Thonny Python, then go to File → Save as…, choose Raspberry Pi Pico, and save your file under the name main.py. You should have by now all electronics and sensors mounted to the SmartHome model and connected to the breadboard.

To connect the OLED display you will need 4 male-to-female jumper wires, and to mount it on the SmartHome model you will need 4 M2 metal bolts and 4 M2 metal nuts.

Wiring diagram for the OLED display:



– connect the red cable to 3v3 rail (+)

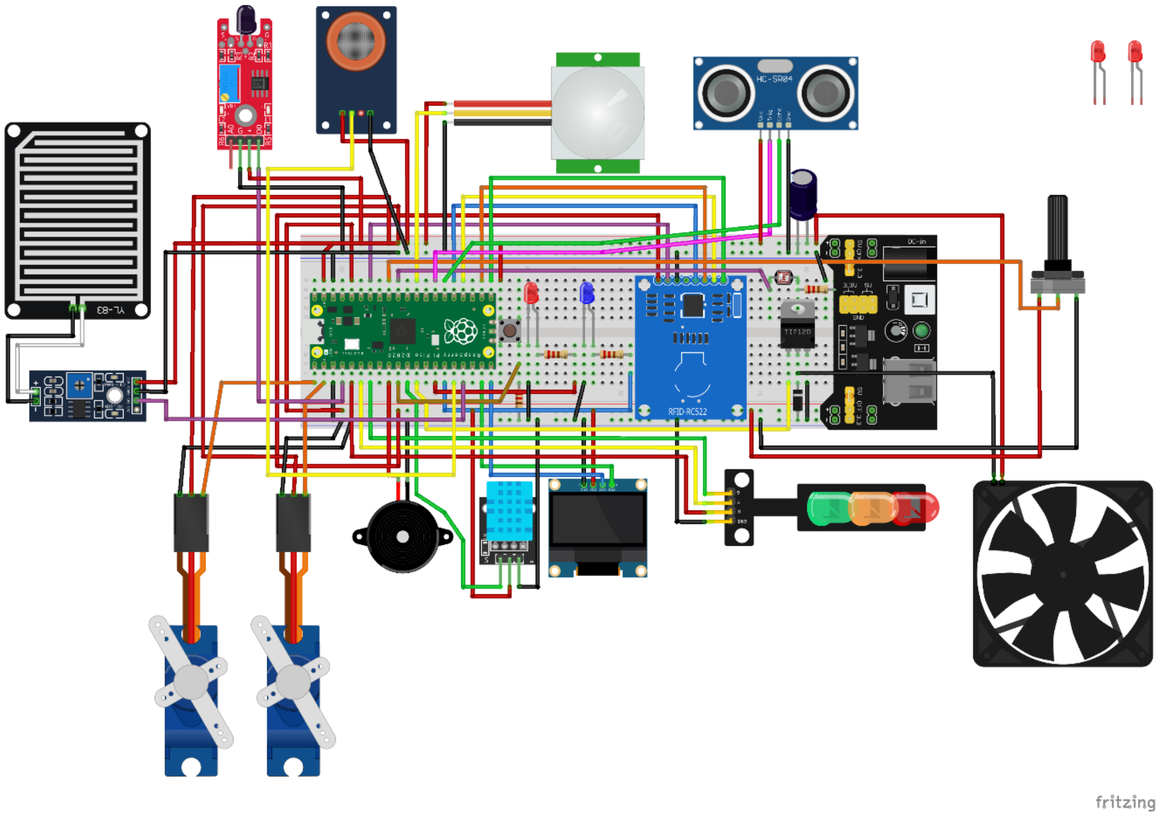
– connect the black cable to GND rail (-)

– connect the green cable to GPIO14 I2C1 SDA pin

– connect the blue cable to GPIO15 I2C1 SCL pin

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Complete wiring diagram for SmartHome4SENIORS Kit:



2.2.3 Write the Micro Python code

The MicroPython code for this tutorial is a compilation of code from the previous tutorials. It contains the setup and control for various components such as pins, sensors, servos, LEDs, buzzers, and displays. It also contains information on how to use the components of the previous tutorials, such as a garage door opener, entrance door security system, doorbell, smart lights, smart thermostat, smart fire alarm, smart air quality detection system, smart rain detection system, and smart security system. The code includes functions that interact with the components and perform actions based on sensor input.

In principle, it should look something like the following:

```
from machine import Pin, ADC, PWM, I2C
from servo import Servo
from hcsr04 import HCSR04
from dht import DHT11
from mfrc522 import MFRC522
from ssd1306 import SSD1306_I2C
from time import sleep

#Define pins for each component
PIN_TRIG = 21
PIN_ECHO = 20
PIN_DOOR = 0
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
PIN_GARAGE = 1
PIN_RED = 3
PIN_YELLOW = 4
PIN_GREEN = 5
PIN_BUZZER = 6
PIN_BUTTON = 7
PIN_PIR = 22
PIN_LED_1 = 10
PIN_LED_2 = 11
PIN_GAS = 12
PIN_RAINDROP = 13
PIN_LDR = 26
PIN_POT = 27
PIN_DHT = 8
PIN_FAN = 9
PIN_FLAME_SENSOR = 2
PIN_SDA = 17
PIN_SCK = 18
PIN_MOSI = 19
PIN_MISO = 16
PIN_RST = 28
PIN_SDA_OLED = 14
PIN_SCK_OLED = 15


#Setup global variables
WIDTH = 128
HEIGHT = 64
MAX = 65535
MID = 32768
MIN = 0

#Setup button
button = Pin(PIN_BUTTON, Pin.IN, Pin.PULL_DOWN)

#Setup buzzer
buzzer = Pin(PIN_BUZZER, Pin.OUT)
buzzer.value(0)

#Setup PIR sensor
PIR = Pin(PIN_PIR, Pin.IN, Pin.PULL_UP)

#Setup LED lights
led_1 = PWM(Pin(PIN_LED_1))
led_2 = PWM(Pin(PIN_LED_2))

led_1.freq(1000)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
led_2.freq(1000)

led_1.duty_u16(MIN)
led_2.duty_u16(MIN)
sleep(5)

#Setup LDR photoresistor
LDR = ADC(Pin(PIN_LDR))
LDR_MAX = 4000
LDR_MIN = 50
LDR_THRESHOLD = int((LDR_MAX + LDR_MIN)/2)

#Setup POT potentiometer
POT = ADC(Pin(PIN_POT))

#Setup for HC-SR04 ultrasonic sensor
ultrasonic = HCSR04(PIN_TRIG, PIN_ECHO)

#Setup for traffic lights module
ledred = Pin(PIN_RED, Pin.OUT)
ledyellow = Pin(PIN_YELLOW, Pin.OUT)
ledgreen = Pin(PIN_GREEN, Pin.OUT)
ledred.value(0)
ledyellow.value(0)
ledgreen.value(0)

#Setup for entrance door
door = Pin(PIN_DOOR)
entrance_door = Servo(pin_id=door)
entrance_door.write(120)  #number should be adjusted depending on
at what angle you have placed the door on the servo motor

#Setup for garage door
garage = Pin(PIN_GARAGE)
garage_door = Servo(pin_id=garage)
garage_door.write(145) #number should be adjusted depending on at
what angle you have placed the garage door on the servo motor

#Setup flame sensor
flame_sensor = Pin(PIN_FLAME_SENSOR, Pin.IN)

#Setup DHT sensor
dht = DHT11(PIN_DHT)

#Setup DC fan
fan = Pin(PIN_FAN, Pin.OUT)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
#Setup RFID Reader RC522
RFID    =    MFRC522(spi_id=0,    sck=PIN_SCK,    miso=PIN_MISO,
mosi=PIN_MOSI, cs=PIN_SDA, rst=PIN_RST)

#Setup for raindrop sensor
rain_sensor = Pin(PIN_RAINDROP, Pin.IN)

#Setup for MQ135 air quality sensor
gas_sensor = Pin(PIN_GAS, Pin.IN)

#Setup for OLED display
i2c = I2C(1, sda=Pin(PIN_SDA_OLED), scl=Pin(PIN_SCK_OLED))
oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
oled.fill(0)

while True:
    #Tutorial 2 - Garage Door
    distance = ultrasonic.distance_cm()
    if distance > 20:
        ledred.value(1)
        ledyellow.value(0)
        ledgreen.value(0)
        garage_door.write(145)
        sleep(0.1)
    elif distance > 10:
        ledred.value(0)
        ledyellow.value(1)
        ledgreen.value(0)
        garage_door.write(145)
        sleep(0.1)
    elif distance > 5:
        ledred.value(0)
        ledyellow.value(0)
        ledgreen.value(1)
        garage_door.write(200)
        sleep(0.1)

    #Tutorial 3 - Entrance Door
    if PIR.value() == 1:
        print("Scan your Key!")
        sleep(1)
    if PIR.value() == 0:
        print("")
        sleep(1)

    RFID.init()
    (stat, tag_type) = RFID.request(RFID.REQIDL)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
    if stat == RFID.OK:
        (stat, uid) = RFID.SelectTagSN()
        if stat == RFID.OK:
            fob = int.from_bytes(bytes(uid),"little",False)

            if fob == 481642800: #change number to your fob ID
                print("Fob ID: "+ str(fob))
                #enable servo motors
                entrance_door.write(200)    #number    should    be
adjusted depending on at what angle you have placed the door on
the servo motor
                sleep(5)
                entrance_door.write(120)
            else:
                print("Fob is not accepted")

    #Tutorial 4 - Doorbell
    if button.value():
        oled.fill(0)
        oled.text("Someone's at the door!", 0, 0)
        buzzer.value(1)
        sleep(0.5)
        buzzer.value(0)
        sleep(1)
        buzzer.value(1)
        sleep(1.5)
        buzzer.value(0)
        sleep(2)

    #Tutorial 5 - Smart Lights
    POT_value = POT.read_u16()
    #print(POT_value)
    led_1.duty_u16(POT_value)
    led_2.duty_u16(POT_value)
    sleep(0.1)

    if PIR.value():
        led_1.duty_u16(MAX)
        led_2.duty_u16(MAX)
        sleep(3)

    LDR_value = LDR.read_u16()
    print (LDR_value)
    if LDR_value > LDR_THRESHOLD:
        led_1.duty_u16(MAX)
        led_2.duty_u16(MAX)
        sleep(0.1)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
    #Tutorial 6 - Smart Thermostat
    dht.measure()
    temp = dht.temperature()
    hum = dht.humidity()

    print("Room Temp:", temp, "°C")
    print("Room Humidity:", hum, "%")

    temperature_threshold = 25 # Adjust the temperature threshold
as needed

    oled.text("Temp: ", temp, 0, 10)
    oled.text("Hum: ", hum, 0, 20)

    if temp > temperature_threshold:
        fan.value(1)
        print("Fan ON")
    else:
        fan.value(0)
        print("Fan OFF")
    sleep(3)

    #Tutorial 7 - Smart Fire Alarm
    if flame_sensor() == 0: #when "fire" is detected
        oled.text("Fire!", 0, 30)
        #buzz the buzzer
        buzzer.value(1)
        #enable servo motors
        entrance_door.write(200)     #number should be adjusted
depending on at what angle you have placed the door on the servo
motor
        garage_door.write(200)   #number should be adjusted
depending on at what angle you have placed the garage door on the
servo motor
        sleep(0.5)  # Wait for 0.5 seconds
    else:
        buzzer.value(0)
        entrance_door.write(120)     #number should be adjusted
depending on at what angle you have placed the door on the servo
motor
        garage_door.write(145)   #number should be adjusted
depending on at what angle you have placed the garage door on the
servo motor
        sleep(0.5)  # Wait for 0.5 seconds

    #Tutorial 8 - Smart Air Quality Detection System
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
if gas_sensor() == 0:
    oled.text("Gas Detected!", 0, 40)
    print("Gas Detected")
    #Activate the buzzer when gas is detected
    buzzer.value(1)
    sleep(0.5)
    #Turn off the buzzer after the delay
    buzzer.value(0)
else:
    print("Gas Not Detected")
    sleep(1.5)

#Tutorial 9 - Smart Rain Detection System
if rain_sensor() == 1:
    print("Not raining")
    sleep(0.5)
else:
    print("It's raining!")
    sleep(0.5)

#Tutorial 10 - Smart Security System
# RFID Reader RC522
print("Bring RFID FOB closer...")
while True:
    RFID.init()
    (stat, tag_type) = RFID.request(RFID.REQIDL)
    if stat == RFID.OK:
        (stat, uid) = RFID.SelectTagSN()
        if stat == RFID.OK:
            fob = int.from_bytes(bytes(uid), "little", False)
            print("FOB ID:", fob)

            if fob == 470550832: #change number to your fob ID
                print("FOB ID accepted")
                oled.text("You may Enter!", 0, 50)
                break
            else:
                print("FOB ID not accepted")

# PIR Motion Sensor
if PIR.value():
    print("Motion detected!")
    buzzer.value(1)
    sleep(0.5)
    buzzer.value(0)
    sleep(0.5)
    buzzer.value(1)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
            sleep(0.5)
            buzzer.value(0)
            sleep(0.5)
            sleep(1)


    oled.show()
```

2.2.4 Application on SmartHome

Now it is time to test your code and circuit on your SmartHome4Seniors house model. Make sure to save the MicroPython code in the Raspberry Pi Pico under main.py name, as instructed at the beginning of the tutorial. Click the Play button in Thonny and bring your SmartHome4Seniors house model to life!

3. Summary

In this tutorial you have learned how you can connect and program an OLED display on your house model to display information about the connected electronics and sensors.

This tutorial utilized all sensors and electronics provided in the SmartHome4SENIORS Kit in order to simulate a fully functional smart home.

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Appendix

ssd1306.py

```
# MicroPython SSD1306 OLED driver, I2C and SPI interfaces

from micropython import const
import framebuf


# register definitions
SET_CONTRAST = const(0x81)
SET_ENTIRE_ON = const(0xA4)
SET_NORM_INV = const(0xA6)
SET_DISP = const(0xAE)
SET_MEM_ADDR = const(0x20)
SET_COL_ADDR = const(0x21)
SET_PAGE_ADDR = const(0x22)
SET_DISP_START_LINE = const(0x40)
SET_SEG_REMAP = const(0xA0)
SET_MUX_RATIO = const(0xA8)
SET_COM_OUT_DIR = const(0xC0)
SET_DISP_OFFSET = const(0xD3)
SET_COM_PIN_CFG = const(0xDA)
SET_DISP_CLK_DIV = const(0xD5)
SET_PRECHARGE = const(0xD9)
SET_VCOM_DESEL = const(0xDB)
SET_CHARGE_PUMP = const(0x8D)

# Subclassing FrameBuffer provides support for graphics primitives
#
http://docs.micropython.org/en/latest/pyboard/library/framebuf.ht
ml
class SSD1306(framebuf.FrameBuffer):
    def __init__(self, width, height, external_vcc):
        self.width = width
        self.height = height
        self.external_vcc = external_vcc
        self.pages = self.height // 8
        self.buffer = bytearray(self.pages * self.width)
        super().__init__(self.buffer,   self.width,   self.height,
framebuf.MONO_VLSB)
        self.init_display()

    def init_display(self):
        for cmd in (
            SET_DISP | 0x00,  # off
            # address setting
            SET_MEM_ADDR,
            0x00,  # horizontal
            # resolution and layout
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
                SET_DISP_START_LINE | 0x00,
                SET_SEG_REMAP | 0x01,  # column addr 127 mapped to SEG0
                SET_MUX_RATIO,
                self.height - 1,
                SET_COM_OUT_DIR | 0x08,  # scan from COM[N] to COM0
                SET_DISP_OFFSET,
                0x00,
                SET_COM_PIN_CFG,
                0x02 if self.width > 2 * self.height else 0x12,
                # timing and driving scheme
                SET_DISP_CLK_DIV,
                0x80,
                SET_PRECHARGE,
                0x22 if self.external_vcc else 0xF1,
                SET_VCOM_DESEL,
                0x30,  # 0.83*Vcc
                # display
                SET_CONTRAST,
                0xFF,  # maximum
                SET_ENTIRE_ON,  # output follows RAM contents
                SET_NORM_INV,  # not inverted
                # charge pump
                SET_CHARGE_PUMP,
                0x10 if self.external_vcc else 0x14,
                SET_DISP | 0x01,
            ):  # on
                self.write_cmd(cmd)
            self.fill(0)
            self.show()

        def poweroff(self):
            self.write_cmd(SET_DISP | 0x00)

        def poweron(self):
            self.write_cmd(SET_DISP | 0x01)

        def contrast(self, contrast):
            self.write_cmd(SET_CONTRAST)
            self.write_cmd(contrast)

        def invert(self, invert):
            self.write_cmd(SET_NORM_INV | (invert & 1))

        def show(self):
            x0 = 0
            x1 = self.width - 1
            if self.width == 64:
                # displays with width of 64 pixels are shifted by 32
                x0 += 32
                x1 += 32
            self.write_cmd(SET_COL_ADDR)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
        self.write_cmd(x0)
        self.write_cmd(x1)
        self.write_cmd(SET_PAGE_ADDR)
        self.write_cmd(0)
        self.write_cmd(self.pages - 1)
        self.write_data(self.buffer)


class SSD1306_I2C(SSD1306):
    def __init__(self, width, height, i2c, addr=0x3C,
external_vcc=False):
        self.i2c = i2c
        self.addr = addr
        self.temp = bytearray(2)
        self.write_list = [b"\x40", None]  # Co=0, D/C#=1
        super().__init__(width, height, external_vcc)

    def write_cmd(self, cmd):
        self.temp[0] = 0x80  # Co=1, D/C#=0
        self.temp[1] = cmd
        self.i2c.writeto(self.addr, self.temp)

    def write_data(self, buf):
        self.write_list[1] = buf
        self.i2c.writevto(self.addr, self.write_list)


class SSD1306_SPI(SSD1306):
    def __init__(self, width, height, spi, dc, res, cs,
external_vcc=False):
        self.rate = 10 * 1024 * 1024
        dc.init(dc.OUT, value=0)
        res.init(res.OUT, value=0)
        cs.init(cs.OUT, value=1)
        self.spi = spi
        self.dc = dc
        self.res = res
        self.cs = cs
        import time

        self.res(1)
        time.sleep_ms(1)
        self.res(0)
        time.sleep_ms(10)
        self.res(1)
        super().__init__(width, height, external_vcc)

    def write_cmd(self, cmd):
        self.spi.init(baudrate=self.rate, polarity=0, phase=0)
        self.cs(1)
        self.dc(0)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
        self.cs(0)
        self.spi.write(bytearray([cmd]))
        self.cs(1)

    def write_data(self, buf):
        self.spi.init(baudrate=self.rate, polarity=0, phase=0)
        self.cs(1)
        self.dc(1)
        self.cs(0)
        self.spi.write(buf)
        self.cs(1)
```

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.