PROJET IOT SMART CORRIDOR

Projet Smartcities

Résumé

Dans ce rapport je vais présenter deux parties de ce projet Smart corridor, la première « IoT sense » qui prend en compte des capteurs de mesure et qui envoie les données vers un serveur via MQTT et la deuxième partie « IoTe » est en effet un éclairage intelligent et qui est également connecté.

HEPL
Haute Ecole de la Province de Liège

Ilya Hasiuk
HEPL 2022 - 2023

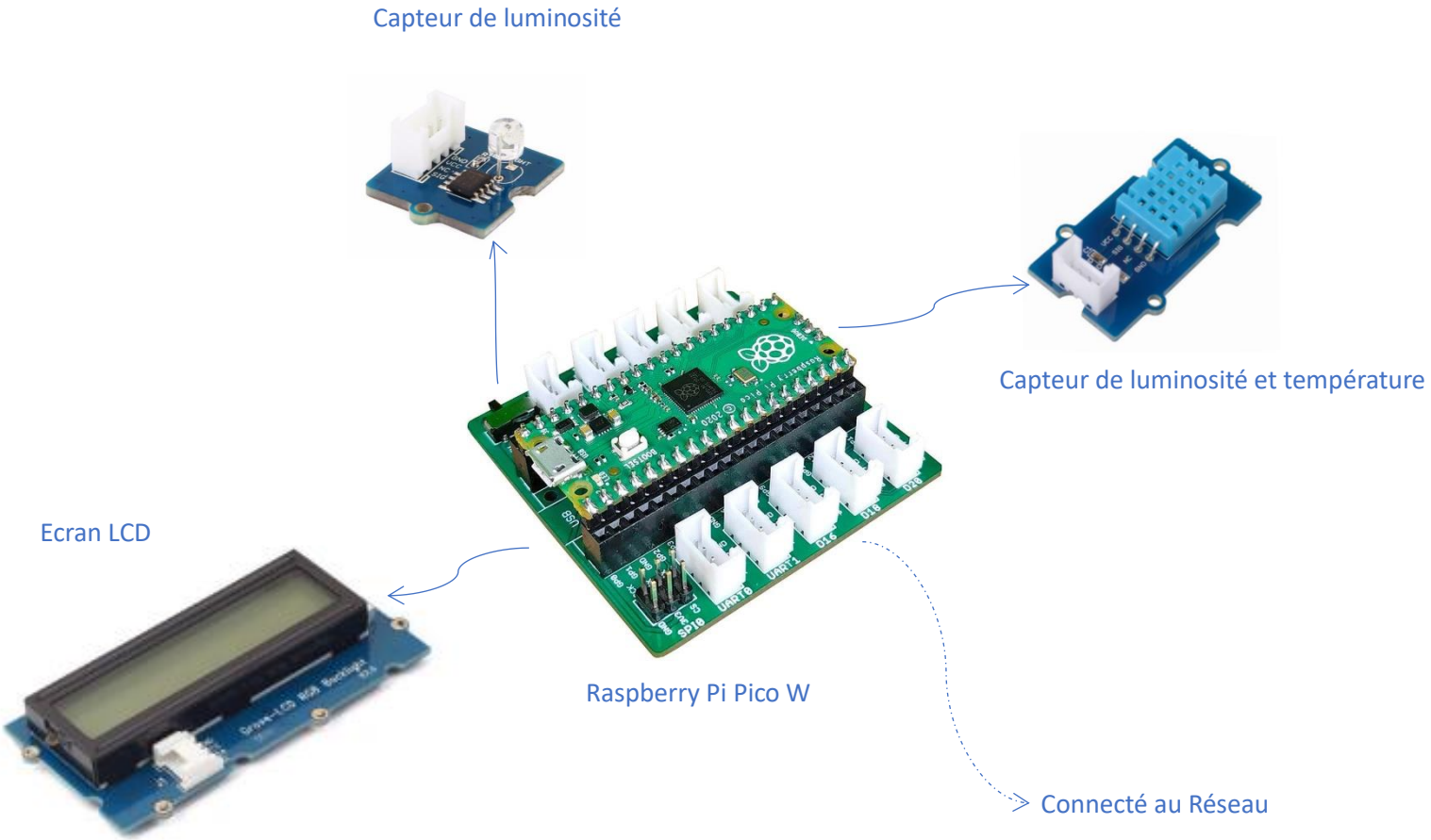
Table des matières

IoT sense

Consignes

- Câblage et choix des borniers.
- Programme en Micro-Python:
    - O Mise à l'heure du RTC via NTP au démarrage.
    - O Boucle infinie:
        - ▪ Lecture de la date et de l'heure sur le RTC, lecture de T, H et luminosité (0 à 100%).
        - ▪ Affichage sur le LCD de la date et de l'heure, de T et de H avec rafraîchissement toutes les 30 s (toutes les secondes pour le test).
        - ▪ Sauvegarde dans un fichier CSV de date, heure, T, H, luminosité.
        - ▪ Envoyer vers un broker MQTT de date, heure, T, H, luminosité.

Matériel nécessaire



Introduction

Dans cette partie réaliser un projet en python qui nous permettra d'assembler des données sur la température, l'humidité et la luminosité, ces données seront stockées dans un fichier .csv sur notre Raspberry et pourront également être envoyés sur le réseau, pour ce projet nous aurons besoin de :

Raspberry Pi Pico 3 :

Le Raspberry Pi Pico W est une carte de développement compacte équipée d'un microcontrôleur ARM Cortex-M0+ 32 bits et d'un module Wi-Fi intégré, conçue par la fondation Raspberry Pi. Elle peut être programmée en utilisant des langages de programmation tels que C/C++ et MicroPython, ce qui la rend idéale pour créer différents projets électroniques et informatiques tels que la domotique, l'automatisation, le contrôle de capteurs et de périphériques, la robotique et la communication sans fil. Avec 26 broches GPIO pour connecter des périphériques électroniques et une mémoire flash de 2 Mo pour stocker le code de programmation et les données, la carte offre une grande polyvalence. En outre, le module Wi-Fi permet une connexion facile à un réseau sans fil, ce qui la rend idéale pour de nombreuses applications.

Kit Groove :

Le kit Grove est un ensemble de modules d'extension électroniques qui peuvent être utilisés avec le Raspberry Pi Pico W pour étendre ses fonctionnalités. Il est conçu pour faciliter la connexion des modules avec le microcontrôleur et pour réduire le temps nécessaire à la mise en place des projets.

Le kit Grove contient différents types de capteurs tels que des capteurs de température, d'humidité, de lumière, de son, de distance, etc. Il contient également des modules d'actionneurs tels que des moteurs, des LED, des relais, etc. Tous ces modules sont compatibles avec le connecteur Grove universel, qui utilise un câble à 4 broches pour la connexion.

En utilisant le kit Grove avec le Raspberry Pi Pico W, il est donc possible de créer rapidement et facilement des projets électroniques avec une grande variété de capteurs et d'actionneurs.

Capteur de lumière :

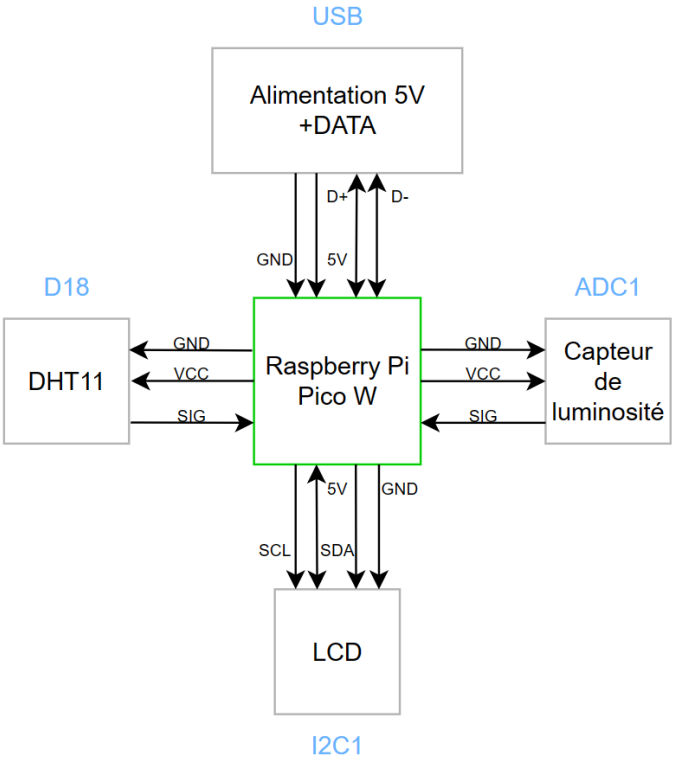
Le capteur de lumière Grove est un module analogique qui peut produire différentes sorties électriques en fonction de la quantité de lumière détectée et qui peuvent être lues par une carte microcontrôleur connectée. C'est un module idéal pour la mesure de la lumière, la détection de la lumière et le contrôle de l'éclairage.

Capteur de luminosité et température :

Le capteur de température et d'humidité Grove est principalement utilisé pour détecter les valeurs de température et d'humidité dans l'environnement. Le capteur de température et d'humidité utilisé initialement était le DHT11. Il a été remplacé plus tard par le DHT20. Veuillez-vous référer à la leçon 10 pour connaître la différence entre les deux.

Mais aussi une connexion au WiFi et de connaissances de Python qui ont été acquises en rédigeant un GitHub avec des différents exercices et divers capteurs : hepl-hasiuk/smartcities (github.com)

Schéma bloc



Connexions

Pour alimenter notre Raspberry Pi Pico W, nous avons choisi d'utiliser l'alimentation USB du PC en 5V. En ce qui concerne les connexions, le capteur de lumière utilise un convertisseur analogique-numérique (ADC) pour convertir le signal analogique en un signal numérique. Un ADC est un composant électronique qui permet de mesurer des signaux analogiques tels que la tension, le courant ou la lumière, et de les convertir en une forme numérique utilisable par un microcontrôleur ou un ordinateur. Dans notre cas, la tension analogique évolue de 0 à +Vcc en fonction de l'intensité lumineuse.

Pour la communication, le LCD utilise une liaison I2C (Inter-Integrated Circuit) qui est un bus de communication série synchrone permettant à plusieurs composants électroniques de communiquer entre eux en utilisant seulement deux fils de données : SDA (Serial Data) et SCL (Serial Clock). Tandis que le DHT11 utilise une liaison digitale, une interface de communication qui utilise des signaux binaires (0 et 1) pour transférer des données entre deux dispositifs électroniques. Cette liaison permet une transmission rapide et fiable de l'information en utilisant une combinaison de niveaux de tension ou de courant pour représenter les différents états logiques.

Le code

```
from machine import Pin
from secrets import *
import network
from lcd1602 import LCD1602
from dht11 import *                                 #Importation of functions and libraries
from machine import Pin,I2C,ADC,PWM,RTC
from utime import sleep, sleep_ms
from umqtt.simple import MQTTClient

i2c=I2C(1,scl=Pin(7), sda=Pin(6), freq=400000)      #define the pins for I2C
d=LCD1602(i2c,2,16)                                 #label data type, number of lines and characters per line
rtc=RTC()                                           #enable acces to the RTC

comp = 0;                                           #define variables
fin = 0;

csv_file = open('Informations.csv', 'w')                                    #Open CSV file for writing
csv_file.write('Date,Time,Temperature (C),Humidity (%),Light (%)\n')        #right the names of the columns

wlan = network.WLAN(network.STA_IF)                                         # Creates a WLAN object and initializes it
wlan.active(True)
wlan.connect(my_secrets["ssid"],my_secrets["WiFi_pass"])

print("Connection to WiFi network.")
print("---------------------------")
print("Trying to connect to WiFi...")
print()

retry = 10                                          # Waits for connection or exit with error code if it fails
while (retry > 0):
    wlan_stat=wlan.status()                         #checks the internet connection
    if wlan_stat==3:
        print("Got IP")
        break
    if wlan_stat==-1:
        raise RuntimeError('WiFi connection failed') #send an error if the connection failed
    if wlan_stat==-2:
        raise RuntimeError('No AP found')
    if wlan_stat==-3:
        raise RuntimeError('Wrong WiFi password')
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    retry = retry-1
    sleep(1)
if wlan_stat!=3:
    raise RuntimeError('WiFi connection failed')

print()
print('Connected to WiFi network: ',end="")
print(wlan.config("ssid"))
print()
ip=wlan.ifconfig()                                  #connection informations
print("IP info (IP address, mask, gateway, DNS):")
print(ip)
print()

monClientMqtt = MQTTClient(client_id=b"p",server=b"192.168.2.135",port=1883)        #creation of the object "monClientMqtt"

monClientMqtt.connect()                                                             #Connection to the broker

while True:
    light = round((light_sensor.read_u16())/40700*100)                              #Calibration based on the maximum received value
    temp, humid=dht.readTempHumid()                                                 #Create values for the humidity and the temperature
    humid = round(humid)                                                            #round thoses values
    temp1 = round(temp)
    dt=rtc.datetime()                                                               #get the date and time with the RTC
    d.clear()                                                                       #clears whats on the screen
    d.setCursor(0,0)                                                                #set the LD cursor
    (year, month, day, weekday, hour, minute, second, subsecond) = rtc.datetime()   #get all the informations wiht RTC

    if comp < 5:                                                                    #if comp is under 5
        d.setCursor(3,1)                                                            #set the cursors and write
        d.print("{:02}/{:02}/{:02}".format(day, month, year))                       #the date and time
        d.setCursor(4,0)
        d.print("{:02}:{:02}:{:02}".format(hour, minute, second))

    else:                                                                           #else
        d.setCursor(4,0)                                                            #set the cursors and write the
        d.print("T:{:.1f}".format(temp))                                            #the temperature
        d.write(0b11011111)                                                         #°
        d.print("C")                                                                #C
        d.setCursor(2,1)
        d.print("H:"+str(humid)+"%")                                                #humidity
        d.setCursor(9,1)
        d.print("L:"+str(light)+"%")                                                #light

    if comp == 29:                                                                  #if compt = 29 (30 sec)
        #write the informations on a csv
        csv_file.write("{:02}/{:02}/{:02},{:02}:{:02}:{:02},{:d},{:d},{:d}\n".format(year, month, day, hour, minute, second, temp1, humid,int(light)))
        csv_file.flush()                                                            #Flush data to file to ensure it is
        comp = 0                                                                    #set comp to 0
        fin = fin +1                                                                #add 1 to fin

    if fin == 120:                                                                  #if fin = 120
        break                                                                       #close the programm
    comp = comp + 1                                                                 #add 1 to comp
    print(fin)                                                                      #print fin, the numbers of
    sleep(1)

                                                                                    #publish the values on the broker
    monClientMqtt.publish("testTopic","{:02}/{:02}/{:02},{:02}:{:02}:{:02},{:d},{:d},{:d}\n".format(year, month, day, hour, minute, second, temp1, humid,int(light)))
csv_file.close()          # Close CSV file when program ends
wlan.disconnect()         # Close connection
```

En pratique

Les données sont bien envoyés vers l'écran LCD, afin d'améliorer l'affichage nous avons décidé d'afficher la date et l'heure pendant 5 secondes et les paramètres de mesures généraux durant 25 secondes suivantes et ainsi de suite.

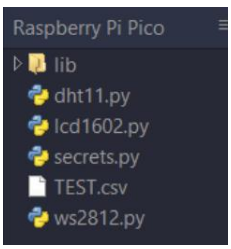
Vidéo avec la démonstration du fonctionnement sur SharePoint :

345303741_6443440845678707_4883486742542472303_n.mp4

Fichier .csv

	A	B	C	D	E
1	Date	Time	Temperature (C)	Humidity (%)	Light (%)
2	10-05-23	13:45:26	27	42	99
3	10-05-23	13:45:56	27	42	55
4	10-05-23	13:46:26	26	46	55
5	10-05-23	13:46:56	26	47	82
6	10-05-23	13:47:26	26	45	98
7	10-05-23	13:47:56	26	45	98
8	10-05-23	13:48:26	26	43	99
9	10-05-23	13:48:56	26	42	99
10	10-05-23	13:49:26	26	43	98
11	10-05-23	13:49:56	26	44	63
12	10-05-23	13:50:26	26	49	62

Ce fichier est créer dans la mémoire du Raspberry PI Pico W, ensuite ce fichier .csv peut être transféré sur notre PC et être ouvert avec EXCEL afin de pouvoir l'analyser et/ou l'intégrer dans une interface graphique. Mais dans notre cas les données sont envoyer vers un Broker MQTT afin que la personne qui reçoit ces données, puisse les utiliser à distance. Pour la facilité les données sont rangées par colonnes.



L’envoie des données cers un Broker MQTT

```
pi@mvb:~/Documents/pythonProjetIot/pythonMqtt/csv $ python pandasReadCsv.py
                    date     heure  température  humidité  luminosité
0   Hello de mon Pico 2W!       NaN          NaN       NaN          NaN
1              2023/05/09  16:50:43         27.0      50.0         33.0
2              2023/05/09  16:50:44         27.0      50.0         42.0
3              2023/05/09  16:50:45         27.0      51.0         42.0
4              2023/05/09  16:50:46         27.0      51.0         37.0
5              2023/05/09  16:50:47         27.0      51.0         39.0
6              2023/05/09  16:50:48         27.0      51.0         42.0
7              2023/05/09  16:50:49         27.0      50.0         33.0
8              2023/05/09  16:50:50         27.0      50.0         51.0
9              2023/05/09  16:50:51         27.0      51.0         40.0
10             2023/05/09  16:50:52         27.0      51.0         56.0
11             2023/05/09  16:50:53         27.0      51.0         50.0
12             2023/05/09  16:50:54         27.0      51.0         55.0
13             2023/05/09  16:50:55         27.0      51.0         53.0
14             2023/05/09  16:50:56         27.0      51.0         51.0
15             2023/05/09  16:50:57         27.0      51.0         56.0
16             2023/05/09  16:50:58         27.0      51.0         47.0
17             2023/05/09  16:50:59         27.0      51.0         56.0
18             2023/05/09  16:51:00         27.0      51.0         42.0
19             2023/05/09  16:51:01         27.0      51.0         51.0
20             2023/05/09  16:51:02         27.0      51.0         41.0
21             2023/05/09  16:51:03         27.0      50.0         46.0
22             2023/05/09  16:51:04         27.0      50.0         42.0
23             2023/05/09  16:51:05         27.0      50.0         56.0
```

Dans cette partie nous pouvons voir que les informations sont bien reçues sur le Broker qui est dans notre cas le Raspberry Pi 3. Pour ce faire nous avons du indiquer l’adresse IP pour l’envoie et le topic sur lequel on publie ces infos, c’est une manière de sécuriser afin qu’elles ne soient pas envoyés à tout le monde.

Conclusion

En conclusion, notre projet a été couronné de succès. Nous avons réussi à mettre en place une synchronisation automatique de notre module RTC avec le serveur NTP, garantissant ainsi la précision des mesures temporelles effectuées par notre système.

Nous avons également créé une boucle de lecture des données des capteurs, permettant l'affichage des informations de température, d'humidité et de luminosité sur un écran LCD rafraîchi toutes les 30 secondes.

Pour faciliter l'analyse des données, nous avons sauvegardé toutes les informations collectées dans un fichier CSV, ces informations peuvent également être envoyées sur le réseau. En somme, nous avons développé un système complet et fonctionnel de mesure du temps et de l'environnement.

Grâce à l'intégration de différents capteurs et à la synchronisation NTP, nos données sont précises et fiables. Toutefois, il est possible d'améliorer ce système en ajoutant d'autres capteurs ou en créant une interface utilisateur plus intuitive pour une utilisation plus pratique dans différents contextes d'application.

IoTe

Consignes

• Câblage et choix des borniers.

• Programme en MicroPython:

O Souscrire à alarme («ON» ou «OFF») sur le broker MQTT

O Boucle infinie:

▪A bsence de présence: luminosité à 10 %, couleur blanche.

▪ Détection de présence: luminosité à 100 %, couleur blanche.

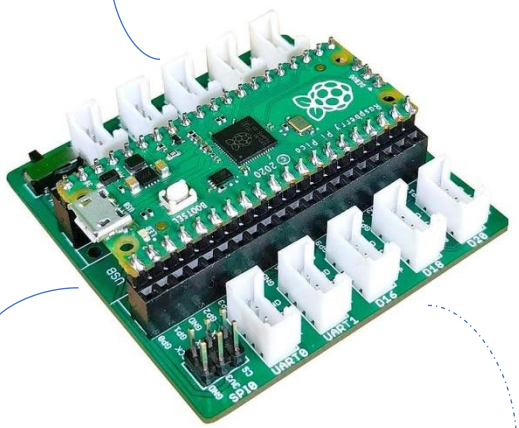
▪ Si alarme = «ON» faire clignoter en rouge avec une luminosité de 100 % les 20 LED du centre du ruban, autres LED en blanc à 100%

▪ Si alarme = «OFF», mode normal.

Matériel nécessaire

Ruban Neopixel (150 LEDs)

Transformateur 220V -> 5V



Raspberry Pi Pico W

Capteur de mouvement

Connecté au Réseau

Introduction

Dans cette deuxième partie nous allons créer un couloir avec un éclairage intelligent, lorsque personne ne passe dans ce couloir la luminosité des LEDs sera à 10% si le capteur de mouvement détectera une présence alors les LEDs seront allumé à 100% afin d'éclairer le passage. Une alarme pourra être enclenchée soit via un bouton poussoir, soit grâce à une connexion au réseau au quel sera connecté notre Raspberry vers lequel on enverra ON ou OFF à partir d'un autre Raspberry Pi 3 pour activer ou désactiver l'alarme. Lorsque cette alarme sera à ON alors les 20 LEDs centrales clignoteront en rouge à 100% de luminosité et toutes les autres seront allumées à 100% en blanc.

Nous allons avoir besoin de :

Capteur de mouvement (PIR) :

Le capteur de mouvement PIR Mini Grove utilise des rayonnements infrarouges passifs pour détecter les mouvements, et est souvent utilisé dans des projets tels que les systèmes d'alarme, les systèmes de visiteurs, les interrupteurs de lumière, etc.

Bouton poussoir :

Le Grove - Bouton est un bouton "momentané marche/arrêt" indépendant. "Momentané" signifie que le bouton rebondit de lui-même après avoir été relâché. Le bouton émet un signal élevé lorsqu'il est enfoncé et un signal bas lorsqu'il est relâché.

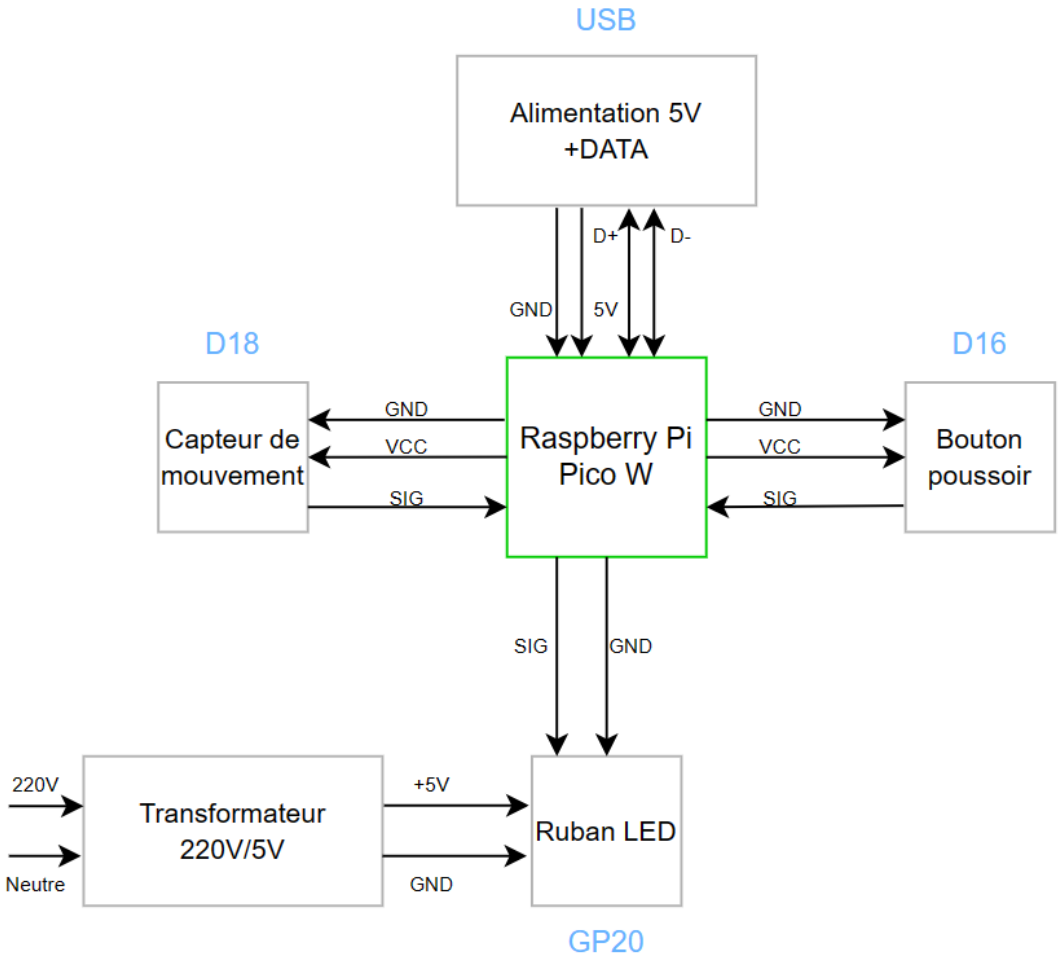
Ruban néopixel :

Un ruban Neopixel, également appelé bande LED Neopixel, est un ruban LED flexible qui contient des LED programmables individuellement. Les LED Neopixel sont souvent utilisées dans les projets de bricolage électronique pour créer des effets lumineux personnalisés, tels que des animations, des motifs, des couleurs changeantes, etc. Les LED sont contrôlées par des microcontrôleurs qui envoient des signaux numériques pour ajuster la couleur, la luminosité et les autres propriétés de chaque LED individuellement.

Transformateur 220V vers 5V

Un transformateur 220V vers 5V est un type de transformateur qui permet de convertir le courant alternatif à haute tension du secteur (220V) en courant continu à basse tension (5V). Cela permet d'alimenter des appareils électroniques qui nécessitent une tension de 5V, tels que les microcontrôleurs, les capteurs et les modules électroniques, ou comme dans notre cas ce sera pour le ruban LED NEOPIXEL, à partir du courant électrique disponible dans les prises murales. Ces transformateurs sont souvent utilisés dans les projets électroniques et informatiques pour fournir une alimentation électrique stable et fiable aux composants électroniques.

Schéma bloc



Connexions

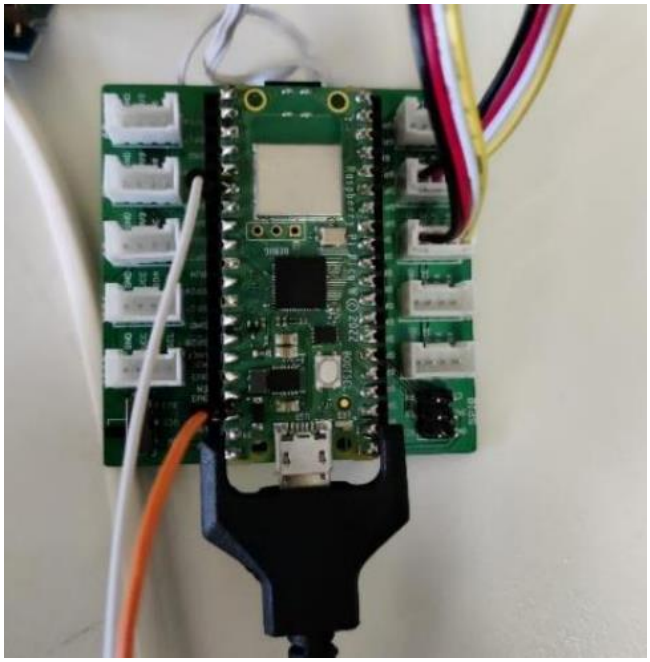
Pour alimenter notre Raspberry Pi Pico W, nous avons opté pour l'utilisation de l'alimentation par USB du PC en 5V. Pour communiquer avec le Raspberry Pi, le PIR et le ruban néopixel utilisent un signal digital via une liaison digitale. Cette interface de communication permet une transmission rapide et fiable de l'information en utilisant des signaux binaires (0 et 1) pour transférer des données entre deux dispositifs électroniques.

Pour contrôler le ruban néopixel, nous utilisons le GPIO 20, qui peut être programmé pour fonctionner en mode entrée ou sortie. En mode entrée, les signaux peuvent être utilisés pour détecter des événements tels que des changements d'état, des impulsions ou des mesures de tension. En mode sortie, il peut être utilisé pour contrôler des périphériques tels que des LED, des moteurs, des capteurs, des relais, etc.

L'alimentation des 5 volts pour le ruban LED est assurée par un transformateur 220/5 volts branché sur une prise. Enfin, notons que le Raspberry Pi est également connecté au réseau. En somme, cette configuration nous permet de contrôler le ruban néopixel à partir de notre Raspberry Pi Pico W de manière efficace et fiable, en utilisant des signaux numériques et un port GPIO programmable.

Branchement physique

Alimentation 220V/5V



Raspberry Pi Pico W branché sur le ruban, capteur de mouvement et le bouton poussoir

Le code

```
from utime import sleep #Importation of functions and libraries
from machine import Pin
from ws2812 import WS2812
import network
import machine
import utime
from simple import MQTTClient

led_count = 150   #numbers of led
mid_start = 65    #index of the first led that have to blink
mid_end = 84      #index of the last led that have to blink

pir= Pin(18,Pin.IN)                                 #set PIR pin number and define it as an output
led = WS2812(20,led_count)                          #WS2812(pin_num,led_count)
BUTTON = machine.Pin(16,machine.Pin.IN)             #set button pin number
#define the colors
red=(255,0,0)
white=(255,255,255)
colors=(red,white) #creation of a tuple with the colors

val = 0                                     #set the value of val to 0

WIFI_SSID = 'electroProjectWifi'                    #enter ssid and password
WIFI_PASSWORD = 'M13#MRSE'

print('Connexion au réseau WiFi...')
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_SSID, WIFI_PASSWORD)
while not wlan.isconnected():
    pass
print('Connecté au réseau WiFi')
                                                    #Waits for connection or exit with error code if it fails
retry = 10
while (retry > 0):
    wlan_stat=wlan.status()                         #checks the internet connection
    if wlan_stat==3:
        print("Got IP")
        break
    if wlan_stat==-1:
        raise RuntimeError('WiFi connection failed')   #send an error if the connection failed
    if wlan_stat==-2:
        raise RuntimeError('No AP found')
    if wlan_stat==-3:
        raise RuntimeError('Wrong WiFi password')
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    retry = retry-1
    sleep(1)

if wlan_stat!=3:
    raise RuntimeError('WiFi connection failed')

print()
print('Connected to WiFi network: ',end="")
print(wlan.config("ssid"))
print()
ip=wlan.ifconfig()                                  #connection informations
print("IP info (IP address, mask, gateway, DNS):")
print(ip)
print()

message=""
received = False

def receivedMQTTMsg(topic,msg):
    global message                                                      #define the callback function
    global received
    message=msg.decode("utf-8")
    received=True

cli = MQTTClient(client_id=b"test",server=b"192.168.2.129",port=1883)           #creation of the object "monClientMqtt"
cli.connect()                                                                   #connection to the broker
cli.set_callback(receivedMQTTMsg)
cli.subscribe("alarm")                                                          #subscribe to alarm
i=0                                                                             #set different variables values
message = 0

val = 0
while True:

    cli.check_msg()                                             #function to receive the message
    if received:                                                #if message new message is recevived, print it
        received = False
        print(message)

    if BUTTON.value() == 1:
        print('1')                                  #if the button value is 1
        val = val + 1                               #increment val by 1
        utime.sleep_ms(500)                         #programm stop for 0.5 sec
```

```
91
92      elif val == 2:                                  #else if val is 2
93           print('2')
94           val = 0
95
96      if message == "ON" or val==1:                                  #if "on" received on the broker or if the button is pressed
97          print("Alarm On")
98
99          led.brightness=1
100         led.pixels_fill(white)                                      #100%
101
102         for i in range(mid_start, mid_end+1):                       #make the led blink red
103             led.pixels_set(i, (255, 0, 0))                          #set the led of the middle on red
104         led.pixels_show()                                           #start the leds
105         sleep(0.2)                                                  #waits 200 ms
106
107         for i in range(mid_start, mid_end+1):
108             led.pixels_set(i, (0, 0, 0))                            #set the leds on black
109         led.pixels_show()                                           #start
110         sleep(0.2)                                                  #waits 200 ms
111
112     else:
113         print("Alarm Off")
114
115         if pir.value()==1:                                          #if motion is detected -> pir = 1
116             led.brightness=1                                        #put the led brightness on 1
117             print('Motion Detected')
118             sleep(0.2)                                              #waits 200 ms
119             led.pixels_fill(white)                                 #set the color on white
120             led.pixels_show()                                      #start the leds
121
122         if pir.value()==0:                                         #if no motion is detected -> pir = 0
123             led.brightness=0.1                                     #set the led brightness on 0.1
124             print('No Detected')
125             sleep(0.2)                                             #waits for 200 ms
126             led.pixels_fill(white)                                 #set the color on white
127             led.pixels_show()                                      #start the leds
```

Les vidéos de démonstrations

Détection de mouvement,alors allumer les LEds à 100% de luminosité en blanc, sinon 10%

https://drive.google.com/file/d/1MAUk0nECFD_x9wnZC8fNShDXr5rl_Em0/view?usp=sharing

Alarm ON et OFF, dabord via le boutton puis via le réseau avec un Raspberry Pi 3 et son invité de commande.

https://drive.google.com/file/d/1HRdHH5QZUdXc8Go4owfW8yrmeWK5VDuQ/view?usp=sharing

https://drive.google.com/file/d/1ddNbOxTMwBZFs_yXR_2IMX2v5vE9itZi/view?usp=sharing

Conclusion

En conclusion, notre projet a été une réussite totale, grâce à la création d'un système performant et efficace pour la souscription à une alarme via le broker MQTT et pour la détection de présence et l'ajustement de la luminosité en fonction de la situation.

Le ruban LED s'avère être un ajout important et utile pour signaler la présence d'une alarme avec des couleurs et des luminosités adaptées, permettant une compréhension claire et précise.

Bien que ce projet puisse être amélioré en ajoutant d'autres fonctionnalités ou en l'adaptant à d'autres utilisations, nous sommes fiers d'avoir réalisé avec succès ce projet et nous sommes impatients d'explorer davantage les possibilités offertes par ce système.