Wprowadzenie do techniki sensorowej - ćwiczenia laboratoryjne		
Sprawozdanie z projektu „Zagadka”		
wykonała Katarzyna Pióro	AGH, Elektronika	Rok 2 / Semestr 4
30.05.2021 r.	I stopnia	Stacjonarne

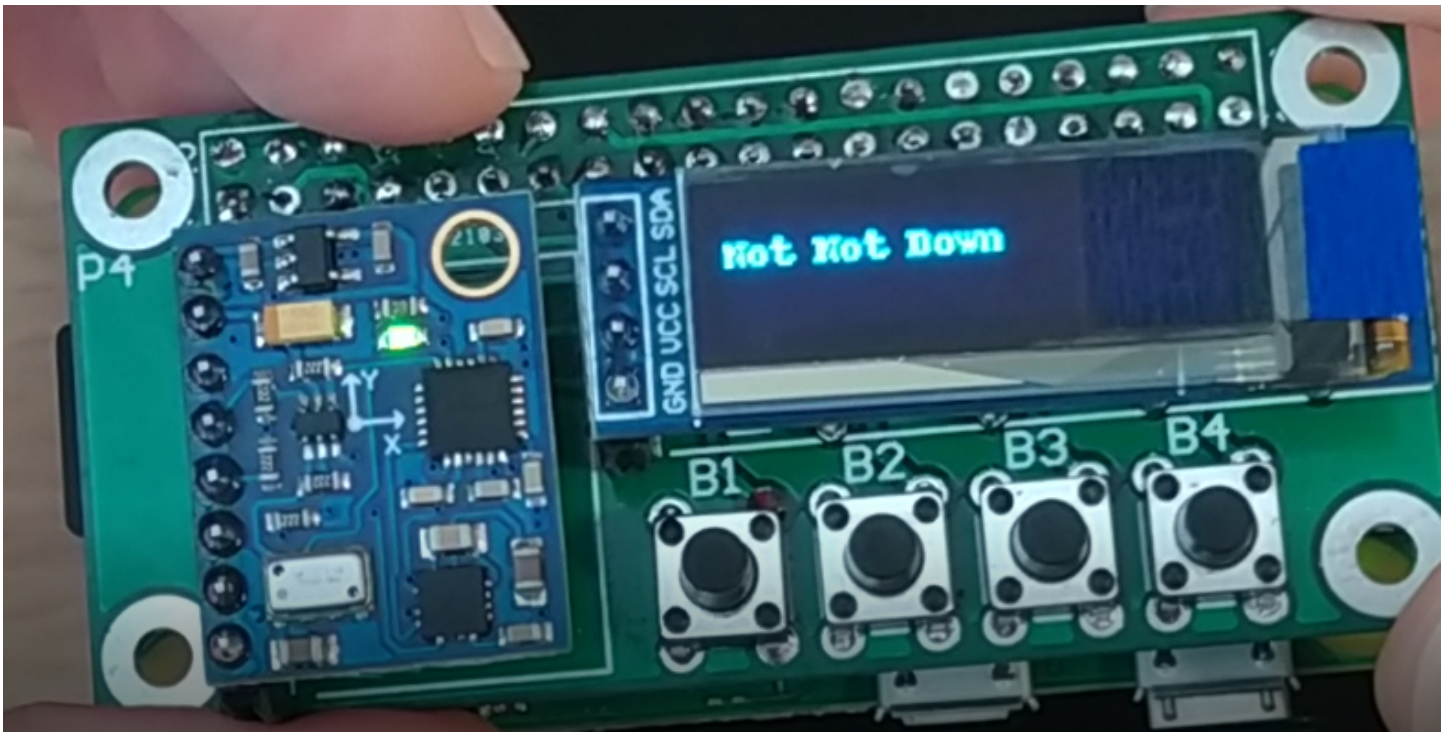
Wstęp

Przedmiotem poniższego sprawozdania jest projekt zrealizowany na minikomputerze Raspberry Pi, przy użyciu języka Python i dodatkowych peryferii podłączonych do płytki Raspberry. Skrypty Gyroscope.py oraz screen.py wchodzące w skład projektu, głównie opierały się na załączonych do Raspberry przykładowych programach, które pokazywały jak odczytać dane z sensorów.

Link do filmu z prezentacją

https://www.youtube.com/watch?v=5gYVdILnknY

Instrukcja



Rysunek 1. Na zdjęciu widać przykładowe polecenie w grze.

“Zagadka” jest prostą grą zręcznościową, w której liczy się czytanie ze zrozumieniem i refleks. Gra polega na odczytywaniu poleceń z ekranu i poruszaniu całym modułem w dwóch osiach - osi x i osi y, co oznacza w praktyce, że całość można wychylić do tyłu, do przodu, na prawo, lub na lewo. Wykonanie ruchu poprawnie pozwala grać dalej, pomyłka natomiast powoduje koniec gry. Gra posiada możliwość włączenia ponownie za pomocą przycisku B1 na lutowanej płytce.

Zastosowane sensory

By monitorować na bieżąco wychylenia urządzenia ograniczono się do zczytywania pomiarów z **żyroskopu**,w taki sposób, żeby reagować na skrajne wartości(które mogą oznaczać wychylenie urządzenia w daną stronę) oraz/lub by sumować wartości wychyleń w danej osi, po to, żeby móc śledzić aktualne położenie. Dostępny żyroskop mierzy tylko zmianę nachylenia kąta przedmiotu w czasie, dlatego, gdyby poruszać powoli to program mógłby nie zaliczyć ruchu np. w prawo, bo zmiana położenia nie była wystarczająco gwałtowna, stąd ważne jest monitorowanie aktualnego położenia.

Możliwe drogi rozwoju

Do gry można dodać więcej, bardziej zawiłych poleceń, które wymagałyby więcej myślenia od użytkownika przed wykonaniem ruchu. Można również dopracować interfejs graficzny, np zrobić pasek dynamicznie skracający się pasek, który będzie pokazywał kończący się czas, jeżeli wahamy się z wykonaniem ruchu. Więcej zawiłych polecen moze wymagać ułozenie nowych kluczy odpowiedzi, tak by był jak najprostszy ale tez jak najbardziej uniwersalny.

Skrypt główny

main.py

```
### Katarzyna Pióro, Elektronika rok 2, AGH, WdTS
### Projekt na Raspberry Pi: Zagadka
from Gyroscope import read_turn
from screen import write_text
from time import sleep  # import
import random
#https://www.raspberrypi.org/documentation/usage/gpio/python/README.md
from gpiozero import LED, Button

#Lista dostępnych opcji
def switchCase(caseNum):
    turns = {
        1: "Up",
        2: "Down",
        3: "Right",
        4: "Left",
        5: "Nothing",
        6: "Do Nothing",
        7: "Not Up",
        8: "Not Down",
        9: "Not Right",
        10: "Not Left",
        11: "Not Nothing",
        12: "Not Not Up",
        13: "Not Not Down",
        14: "Not Not Right",
        15: "Not Not Left",
        16: "Not Not Nothing",
        17: "Whatever"
    }
    return turns.get(caseNum)

#Klucz do ułożenia kodu odpowiedzi
def findCorrect(display):
    turns = {
        "Up": "Up",
        "Down": "Down",
        "Right": "Right",
```

```
        "Left": "Left",
        "Nothing": "Nothing",
        "Do Nothing": "Nothing",
        "Not Up": "Up", ##WYMAGA ZAPRZECZENIA
        "Not Down": "Down", ##WYMAGA ZAPRZECZENIA
        "Not Right": "Right", ##WYMAGA ZAPRZECZENIA
        "Not Left": "Left", ##WYMAGA ZAPRZECZENIA
        "Not Nothing": "Nothing", ##WYMAGA ZAPRZECZENIA
        "Not Not Up": "Up",
        "Not Not Down": "Down",
        "Not Not Right": "Right",
        "Not Not Left": "Left",
        "Not Not Nothing": "Nothing",
        "Whatever": "Nothing"  ##WYMAGA ZAPRZECZENIA
    }
    return turns.get(display)

#Świecenie diod
def LEDs(int):
    led_r = []
    led_g = []
    led_r.append(LED(17)) #czerwone
    led_r.append(LED(27)) #czerwone
    led_g.append(LED(22)) #zielone
    led_g.append(LED(23)) #zielone
    j=0
    while j<9000:
        for i in range(2):
            if(int==1):
                led_g[i].on()
            elif(int==2):
                led_r[i].on()
        j=j+1
    led_g[i].off()
    led_r[i].off()


negative = ["Not Up", "Not Down", "Not Right", "Not Left", "Not Nothing",
"Whatever"] #Polecenia dla których inny jest klucz/zasada odczytywania kodu
randNum = []
randomArray_display = []
randomArray_code = []
button = []
restart = True
#---------------------------------Nowa gra---------------------------------
#Nieskończona pętla dopóki użytkownik nie odmówi zagrania w kolejną grę
while(restart==True):
    for j in range(8):
        randNum.append(random.randint(1,17)) #generowanie losowego kodu
        randomArray_display.append(switchCase(randNum[j])) #odczytanie polecen
do gry wyświetlanych na ekranie przypisanych do liczb
        randomArray_code.append(findCorrect(randomArray_display[j]))
#stworzenie kodu poprawnych odpowiedzi


    i = 0
    stop1=False
    dobrze=False

    #Reagowanie na czynności użytkownika i porównywanie ich z kodem
    while(stop1==False):
        write_text("", 0, True) #czyszczenie ekranu
```

```
        write_text(randomArray_display[i], 8, True)

        ruch = read_turn()

        ## Porównywanie wykonanego ruchu z kodem
        #klucz1 do kodu
        if (randomArray_display[i] == randomArray_code[i]): #czy display i code
sa takie same
            if (ruch == randomArray_code[i]): #TAK - kod i ruch użytkownika
mają taką samą nazwę
                dobrze = True
            else:
                dobrze = False #Wykonano niepoprawny ruch
                # koniec
        else: # NIE - sprawdzmy czy klucz do kodu sie zmieni
            stop2 = False
            for k in range(5):
                if(randomArray_display[i] == negative[k]):#czy wyświetlane
polecenie należy do "podchwytliwych"
                    # klucz2 do kodu
                    if (ruch != randomArray_code[i]):  # TAK - podchywytliwe
polecenie: wykonano jakikolwiek ruch oprócz zapisanego w kodzie
                        dobrze = True
                        stop2 = True
                        break
            if (stop2 != True) and (ruch == randomArray_code[i]): # NIE -
polecenie nie należy do "podchwytliwych", polecenie powinno byc identyczne jak
kod
                # klucz1 do kodu
                dobrze = True
                # koniec
            elif (stop2 != True): #Wykonano niepoprawny ruch
                dobrze = False
                # koniec

        if dobrze==True:
            write_text("Correct!", 16, True)
            LEDs(1) #świecenie diod zielonych /Ciąg dalszy
            #grasz dalej
        else:
            write_text("Game over", 16, True) #świecenie diod czerwonych
/Przegrana
            LEDs(2)
            stop1 = True
            break

        #Jeśli nie przegrano, to sprawdź warunek zakończenia
        if i==7: #Maksymalnie 8 prób w ciagu jednej gry
            break
        else:
            i=i+1


    #Czy chcesz zagrać ponownie?
    write_text("Play again?", 8, False)
    sleep(1)
    write_text("B1=Yes, B4=No", 16, False)
    sleep(2)
    #Przypisanie przycisków na listę zmiennych
    button.append(Button(6))
    button.append(Button(13))
    button.append(Button(19))
```

```
    button.append(Button(26))
    #Nieskonczona pętla, ktora moze zostać przerwana tylko przycisnięciem
odpowiedniego przycisku
    while True:
        if button[0].is_pressed: #przycisk B1
            restart = True
            break
        elif button[3].is_pressed: #przycisk B4
            restart = False
            write_text(" ", 0, True)  # czysczenie ekranu
            break
```

Skrypt dodatkowy 1

screen.py

```
import time

import Adafruit_SSD1306

from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

import subprocess



# Copyright (c) 2017 Adafruit Industries
# Author: Tony DiCola & James DeVito
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to
deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.



# Write
#Wypisanie tekstu na ekranie OLED
def write_text(word, y,black):
    # Raspberry Pi pin configuration:
```

```
    RST = None  # on the PiOLED this pin isnt used

    # 128x32 display with hardware I2C:
    disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)

    # 128x64 display with hardware I2C:
    # disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)

    # Note you can change the I2C address by passing an i2c_address parameter
like:
    # disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST, i2c_address=0x3C)

    # Initialize library.
    disp.begin()

    # Clear display.
    disp.clear()
    disp.display()

    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Draw a black filled box to clear the image.
    #Dodatkowy warunek, by móc określić, czy chcemy zetrzeć zawartość
poprzedniego ekranu czy nie
    if(black==True):
        draw.rectangle((0, 0, width, height), outline=0, fill=0)

    # Draw some shapes.
    # First define some constants to allow easy resizing of shapes.
    padding = -2
    top = padding
    bottom = height - padding
    # Move left to right keeping track of the current x position for drawing
shapes.
    x = 0

    # Load default font.
    font = ImageFont.load_default()

    # Alternatively load a TTF font.  Make sure the .ttf font file is in the
same directory as the python script!
    # Some other nice fonts to try: http://www.dafont.com/bitmap.php
    # font = ImageFont.truetype('Minecraftia.ttf', 8)

    #Położenie, treść, czcionka i kolor tekstu
    draw.text((x,top+y), word, font=font, fill=255)

    # Display image.
    disp.image(image)
    disp.display()
```

Skrypt dodatkowy 2

Gyroscope.py

```
import smbus  # import SMBus module of I2C
from time import sleep  # import
import math
from screen import write_text



# some MPU6050 Registers and their Address
PWR_MGMT_1 = 0x6B
SMPLRT_DIV = 0x19
CONFIG = 0x1A
GYRO_CONFIG = 0x1B
INT_ENABLE = 0x38
GYRO_XOUT_H = 0x43
GYRO_YOUT_H = 0x45
GYRO_ZOUT_H = 0x47
Device_Address_MPU = 0x68  # MPU6050 device address


def MPU_Init():
    # write to sample rate register
    bus.write_byte_data(Device_Address_MPU, SMPLRT_DIV, 7)

    # Write to power management register
    bus.write_byte_data(Device_Address_MPU, PWR_MGMT_1, 1)

    # Write to Configuration register
    bus.write_byte_data(Device_Address_MPU, CONFIG, 0)

    # Write to Gyro configuration register
    bus.write_byte_data(Device_Address_MPU, GYRO_CONFIG, 24)

    # Write to interrupt enable register
    bus.write_byte_data(Device_Address_MPU, INT_ENABLE, 1)


def read_raw_data_MPU(addr):
    # Accelero and Gyro value are 16-bit
    high = bus.read_byte_data(Device_Address_MPU, addr)
    low = bus.read_byte_data(Device_Address_MPU, addr + 1)

    # concatenate higher and lower value
    value = ((high << 8) | low)

    # to get signed value from mpu6050
    if (value > 32768):
        value = value - 65536
    return value

#odczyt zmiany położenia kątowego modelu w osi x i osi y
def read_turn():
    Gx_current = 0
    Gy_current = 0
    stop = False
    text = ""

    for i in range(40):
```

```
        # Read Gyroscope raw value
        gyro_x = read_raw_data_MPU(GYRO_XOUT_H)
        gyro_y = read_raw_data_MPU(GYRO_YOUT_H)
        gyro_z = read_raw_data_MPU(GYRO_ZOUT_H)

        # Full scale range +/- 250 degree/C as per sensitivity scale factor
        Gx = gyro_x / 131.0
        Gy = gyro_y / 131.0
        Gz = gyro_z / 131.0

        #aktualizacja maksymalnego dokonanego wychylenia w danej osi
        Gx_current = Gx_current + Gx
        Gy_current = Gy_current + Gy

        #pomocnicze printy do podglądu na żywo na komputerze:
        # print("Gx=%.2f" % Gx, u'\u00b0' + "/s", "\tGy=%.2f" % Gy, u'\u00b0' +
"/s", "\tGz=%.2f" % Gz, u'\u00b0' + "/s")
        #print("Gx_curr:", int(Gx_current), "Gy_curr:", int(Gy_current))

        #Warunki określające położenie
        if (Gx >= 50) or (Gx_current >= 50):
            text = "Down"
            stop = True
        elif (Gx <= -50) or (Gx_current <= -50):
            text = "Up"
            stop = True
        elif (Gy >= 50) or (Gy_current >= 50):
            text = "Right"
            stop = True
        elif (Gy <= -50) or (Gy_current <= -50):
            text = "Left"
            stop = True
        else:
            text = "Nothing"

        if stop == False:
            sleep(0.1)
        else:
            break
    # koniec petli
    return text

#Main code
bus = smbus.SMBus(1)  # or bus = smbus.SMBus(0) for older version boards

MPU_Init()
```