Національний Технічний Університет України

“Київський Політехнічний Інститут”

Факультет Інформатики та Обчислювальної Техніки

Кафедра обчислювальної техніки

**Проектна робота**

з курсу «Системи обробки сигналів»

Додаткова клавіатура з універсальними кнопками.

Виконали:

студенти 3 курсу ФІОТ

групи ІВ-81

Зубець М.В., Бойківський А.О.

Перевірив:

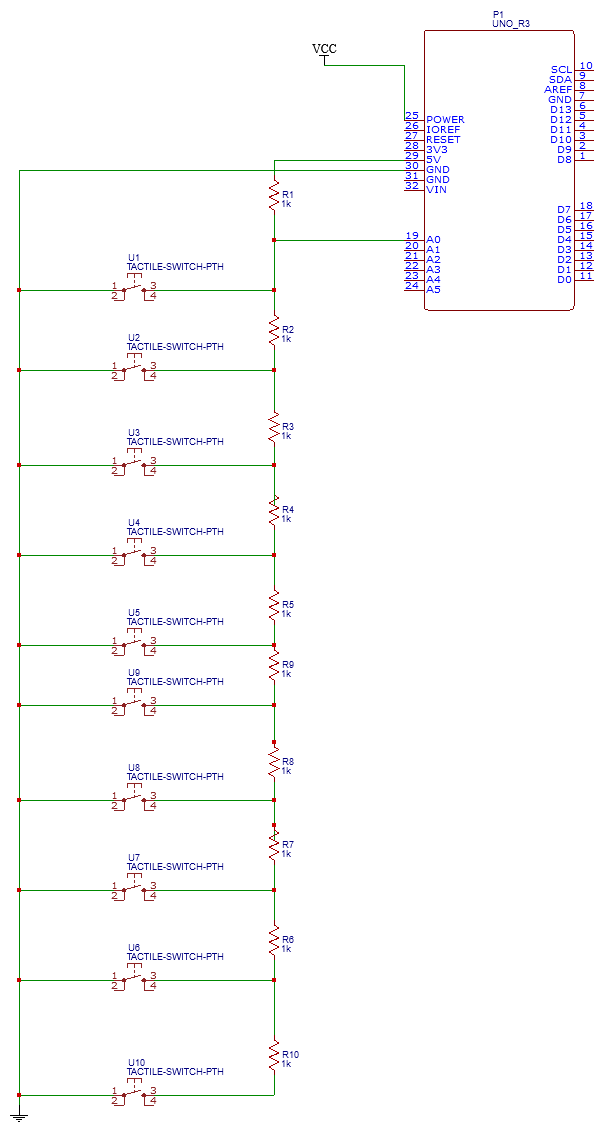
Виноградов Ю.М.

Київ 2020

**Мета:** Розробка пристрою, що містить в собі кнопки, які можна запрограмувати на виконання різних функцій. Приклад функцій – множинне натиснення клавіш, виклик системних ресурсів через консоль.

**Використані ресурси:**

|  |  |  |
| --- | --- | --- |
|  | Arduino UNO R3 (CH340) | 114 грн. |
|  | Макетная плата MB-102 830 отверстий  x2 | 56x2=112 грн. |
|  | Набор перемычек (папа-папа) 40шт 20см | 40 грн. |
|  | Комплект кнопок 12мм с колпачком (6 цветов)x2 | 18x2=36 грн. |
|  | Резистор 1 КОм (5 штук)x3 | 2x3=6 грн. |

**Розроблена схема:**

**Опис схеми:** Інформацію про натискання кнопок реалізовано за рахунок зміни аналогового значення на одному з входів. Аналогове значення змінюється за рахунок зміни напруги через замикання кола кнопками, що підключені паралельно.

**Стек технологій:**

**Python:** arduino-python3, pyautogui, pyQt5

**Arduino Software (IDE)**

**Програмна реалізація:**

*Повний код проекту можна знайти за посиланням:*

[*https://github.com/MkZb/Arduino\_Keyboard*](https://github.com/MkZb/Arduino_Keyboard)

На Arduino встановлено Python Arduino Command API:

<https://github.com/thearn/Python-Arduino-Command-API>

Основний файл обробщик:

from Arduino import Arduino  
import time, sys  
from GUI import \*  
import threading  
  
  
def debounce(wait):  
 *""" Decorator that will postpone a functions  
 execution until after wait seconds  
 have elapsed since the last time it was invoked. """* def decorator(fn):  
 def debounced(\*args, \*\*kwargs):  
 def call\_it():  
 fn(\*args, \*\*kwargs)  
  
 try:  
 debounced.t.cancel()  
 except(AttributeError):  
 pass  
 debounced.t = threading.Timer(wait, call\_it)  
 debounced.t.start()  
  
 return debounced  
  
 return decorator  
  
  
*# debounce decorator makes our function to be executed after button release  
# that allows us to only execute function once even if button is being held*@debounce(0.3)  
def keyPressed(num):  
 window.funct.func\_list[num - 1](num)  
  
  
*# This function for example will be executed every time we listen to the new  
# analog value*def test\_func():  
 print(**"YEP holding btn"**)  
  
  
def listen(pin: int):  
 pin\_analog\_value = board.analogRead(pin)  
 *# handle = open('test.txt', 'r')  
 # pin\_analog\_value = int(handle.read())  
 # handle.close()* **"""  
 BTN LAYOUT:  
 5 4 3 2 1  
 6 7 8 9 10  
 """** if pin\_analog\_value < 10:  
 keyPressed(5)  
 elif pin\_analog\_value < 516:  
 keyPressed(4)  
 elif pin\_analog\_value < 686:  
 keyPressed(3)  
 elif pin\_analog\_value < 771:  
 keyPressed(2)  
 elif pin\_analog\_value < 823:  
 keyPressed(1)  
 elif pin\_analog\_value < 856:  
 keyPressed(6)  
 elif pin\_analog\_value < 881:  
 keyPressed(7)  
 elif pin\_analog\_value < 899:  
 keyPressed(8)  
 elif pin\_analog\_value < 913:  
 keyPressed(9)  
 elif pin\_analog\_value < 924:  
 keyPressed(10)  
  
  
class ListenArduino(threading.Thread):  
 def \_\_init\_\_(self):  
 threading.Thread.\_\_init\_\_(self)  
  
 def run(self):  
 while (True):  
 listen(0)  
 time.sleep(0.2) *# change sleep for different frequency calls*if \_\_name\_\_ == **'\_\_main\_\_'**:  
 board = Arduino(**"115200"**)  
 app = QApplication(sys.argv)  
 window = MainWindow()  
 thread1 = ListenArduino()  
 thread1.start()  
 window.show()  
 app.exec\_()

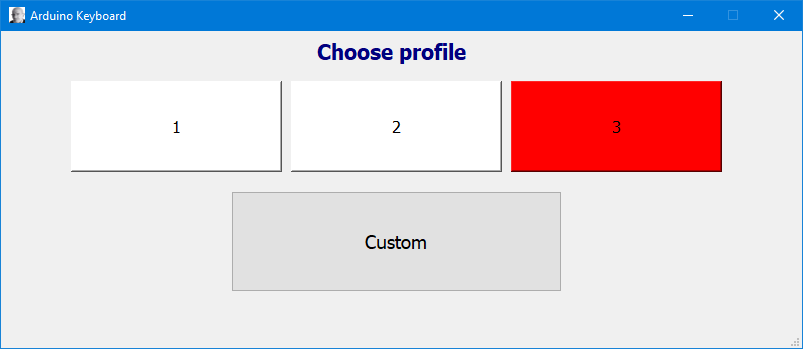
Файл реалізації функціональності:

import pyautogui, os  
  
  
class Functionality():  
  
 def \_\_init\_\_(self, profile=None):  
 self.profile = profile  
 self.func\_list = []  
 self.hotkeys = dict.fromkeys([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], **""**)  
 if profile == 1:  
 for i in range(10):  
 self.func\_list.append(self.pressHotkey)  
 self.setHotkey(i + 1, [**"Ctrl"**, str(i)])  
 elif profile == 2:  
 for i in range(10):  
 self.func\_list.append(self.pressHotkey)  
 self.hotkeys = {1: [**"Ctrl"**, **"c"**],  
 2: [**"Ctrl"**, **"v"**],  
 3: [**"Ctrl"**, **"a"**],  
 4: [**"Ctrl"**, **"shift"**, **"esc"**],  
 5: **"volumeup"**,  
 6: **"volumedown"**,  
 7: **"volumemute"**,  
 8: [**""**, **""**],  
 9: **""**,  
 10: **""**}  
 elif profile == 3:  
 for i in range(10):  
 self.func\_list.append(self.executeCommand)  
 self.hotkeys = {1: **"calc"**,  
 2: **"charmap"**,  
 3: **"control"**,  
 4: **"devmgmt.msc"**,  
 5: **"explorer"**,  
 6: **"mmsys.cpl"**,  
 7: **"notepad"**,  
 8: **"taskmgr"**,  
 9: **"compmgmt.msc"**,  
 10: **"appwiz.cpl"**}  
  
 def setHotkey(self, num, keys):  
 self.hotkeys[num] = keys  
  
 def pressHotkey(self, num):  
 hotkey = self.hotkeys[num]  
 if type(hotkey) is list:  
 pyautogui.hotkey(\*hotkey)  
 else:  
 pyautogui.press(hotkey)  
  
 def setCommand(self, num, command):  
 self.hotkeys[num] = command  
  
 def executeCommand(self, num):  
 command = self.hotkeys[num]  
 print(os.system(command))

**Готовий пристрій:**

****

**Інтерфейс користувача:**

****

**Результати тестування:** Роботу пристрою було перевірено з різними розкладками функціоналу прив’язаних кнопок, та з різним часом затримки опитування плати. В усіх випадках пристрій показав очікувану поведінку, та правильне виконання всіх функцій.

**Висновки:** В ході виконання роботи було розроблено пристрій, що відповідає поставленій меті.

Плюсами обраного шляху реалізації є:

1. Простота реалізції, тобто можливість додати, або прибрати довільну кількість кнопок.
2. Підключення усіх кнопок до 1 аналогового входу, що дозволяє розробити систему, яка містить безліч кнопок як інтерфейс та додаванням безлічі інших пристроїв, за рахунок економії використаних входів.
3. Використання Python API, що дозволило реалізувати інтерфейс користувача методами бібліотек Python, а також додавання багатопоточності.

З мінусів можна виділити:

1. Виконання на макетній платі призвело до незручності у практичному використанні.

Шлях усунення:

Виконати збірку на паяльній платі, що призведе до компактного розташування провідників та резисторів і дозволить розмістити пристрій в корпус для практичного використання.

1. Через підключення усіх кнопок до аналогового входу неможливо відстежити множинне натискання кнопок на пристрої.

Шляхи усунення:

А) Підключення кожної кнопки окремо до різних аналогових входів (неможливо через відсутність необхідної кількості аналогових входів на платі)

Б) Створити схему, в якій значення напруг для всіх кнопок і їх комбінацій були б унікальними. (майже неможливо через велику кількість комбінацій)

1. Відсутність ручного налаштування призначення клавіш. (Реалізована програмна зміна лейаутів)