|  |  |  |  |
| --- | --- | --- | --- |
| **Laboratorium z przedmiotu Systemy wbudowane (SW)** | | | |
|  | | | |
| Karta projektu – zadanie 7 | | | |
|  | | | |
| *Nazwa projektu:*  Budzik | | | |
|  | | | |
| *Prowadzący:*  *Dr. Przemysław Zakrzewski* | *Autorzy (tylko nr indeksu):*  132220  132302  132328 | *Grupa dziekańska:* | *I2 śr 9:45* |
| ***Ocena:*** |  |
|  | | | |
| ***Cel projektu:***  Zaprojektowanie budzika, który będzie wyłączany po naciśnięciu przycisków w kolejności określanej poprzez zaświecenie diod.  Wykorzystanie modułu RTC do przechowywania aktualnego czasu.  Stworzenie bazy przechowującej aktualnie zapisane budziki.  Stworzenie graficznego interfejsu, za pomocą którego użytkownik będzie mógł zalogować się do systemu i ustawić alarm / przeglądać istniejące budziki. | | | |
|  | | | |
| ***Schemat:*** | | | |
|  | | | |
| ***Wykorzystana platforma sprzętowa, czujniki pomiarowe, elementy wykonawcze:***  Projekt zrealizowano przy pomocy Beaglebone Black. Użyte elementy: głośnik, moduł RTC, 4 diody,  4 przyciski, 4 rezystory 1000 omów, płytka uniwersalna. | | | |

***Najważniejsze fragmenty kodu:***

**Implementacja modułu RTC:**

\_REG\_MINUTES = 0x01

\_REG\_HOURS = 0x02

#odczytuje minut i dekodowanie wyniku z formatu 2x4bit BCD na typ integer (\_bcd\_to\_int)

def \_read\_minutes(self):

return \_bcd\_to\_int(self.\_read(self.\_REG\_MINUTES))

#odczytuje godzin i dekodowanie wyniku z formatu 2x4bit BCD na typ integer (\_bcd\_to\_int)

def \_read\_hours(self):

d = self.\_read(self.\_REG\_HOURS)

if d & 0x40:

return \_bcd\_to\_int(d & 0x3F)

else:

h = \_bcd\_to\_int(d & 0x1F)

if d & 0x20:

h += 11 # Convert 12h to 24h

elif h == 12:

h = 0

return h

**Włączanie głośnika:**

PWM.start("P8\_13", 25, 1000)

**Przykładowy kod powodujący zaświecenie się pierwszej diody i zgaszenie jej po naciśnięciu odpowiedniego przycisku. Analogicznie jest dla pozostałych diod i przycisków.**

button = 1

if rand == 1: #rand-wylosowano liczbę od 1 do 4 określającą, która dioda zostanie zaświecona

GPIO.output("P8\_7", GPIO.HIGH) #zaświeca pierwszą diodę

while button == 1: #tak długo, aż nie naciśnięto przycisku

button = GPIO.input("P8\_8") #sprawdza, czy naciśnięto przycisk

while button == 0: #tak długo jak przycisk jest naciśnięty

button = GPIO.input("P8\_8")

GPIO.output("P8\_7", GPIO.LOW) #powoduje zgaszenie diody

**Łączenie z bazą**

DATABASE = 'baza.db'

db = g.\_database = sqlite3.connect(DATABASE)

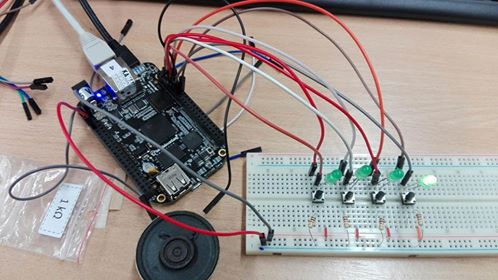
cursor = db.cursor()

cursor.execute('CREATE TABLE IF NOT EXISTS alarmy(id INT PRIMARY KEY,user TEXT, godzina INT, minuty INT);')

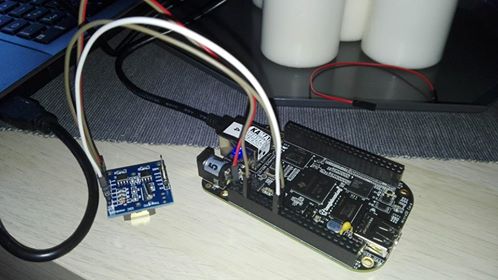
***Projekt a realizacja:***

Podczas realizacji projektu zastąpiliśmy głośnik Bluetooth głośnikiem okrągłym dostępnym   
w laboratorium. Poza tym zrealizowaliśmy wszystkie założenia projektu. Projekt można by rozwinąć poprzez dodanie wyświetlacza pokazującego aktualny czas (w naszej wersji projektu pokazuje   
w aplikacji).

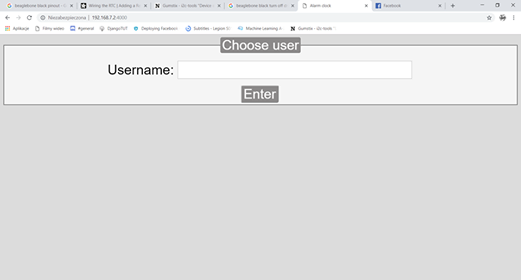
***Zdjęcie połączenia:***

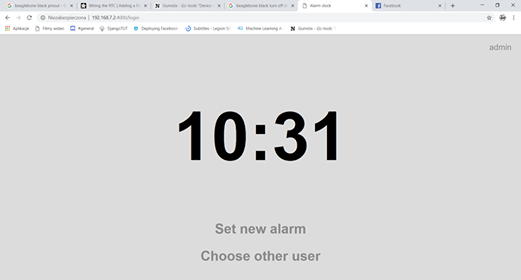
******

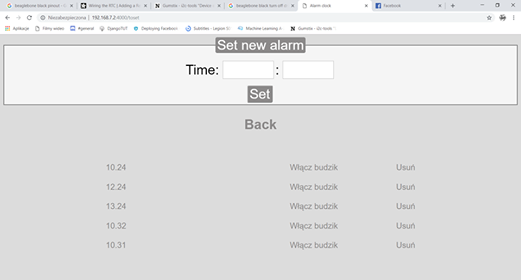
Moduł RTC:

******

***Zrzuty ekranu z aplikacji:***

******

******

******

***Podsumowanie:***

Projekt realizuje wyznaczone cele. Podczas jego wykonywania poszerzyliśmy swoją wiedzę o takie elementy jak np. zasadę działania modułu RTC. Ogólnie był to ciekawy sposób wykorzystania wiedzy do stworzenia produktu mającego zastosowanie w codziennym życiu.