

**Internet Rzeczy – Projekt “Smart Drink Dispenser”: Raport 2**

Prowadząca: mgr inż. Aleksandra Wilczyńska

**Inżynierskie Zastosowania Informatyki w Elektrotechnice**

IZI2 1.1/1

**Wykonanie**

Cichocki Daniel

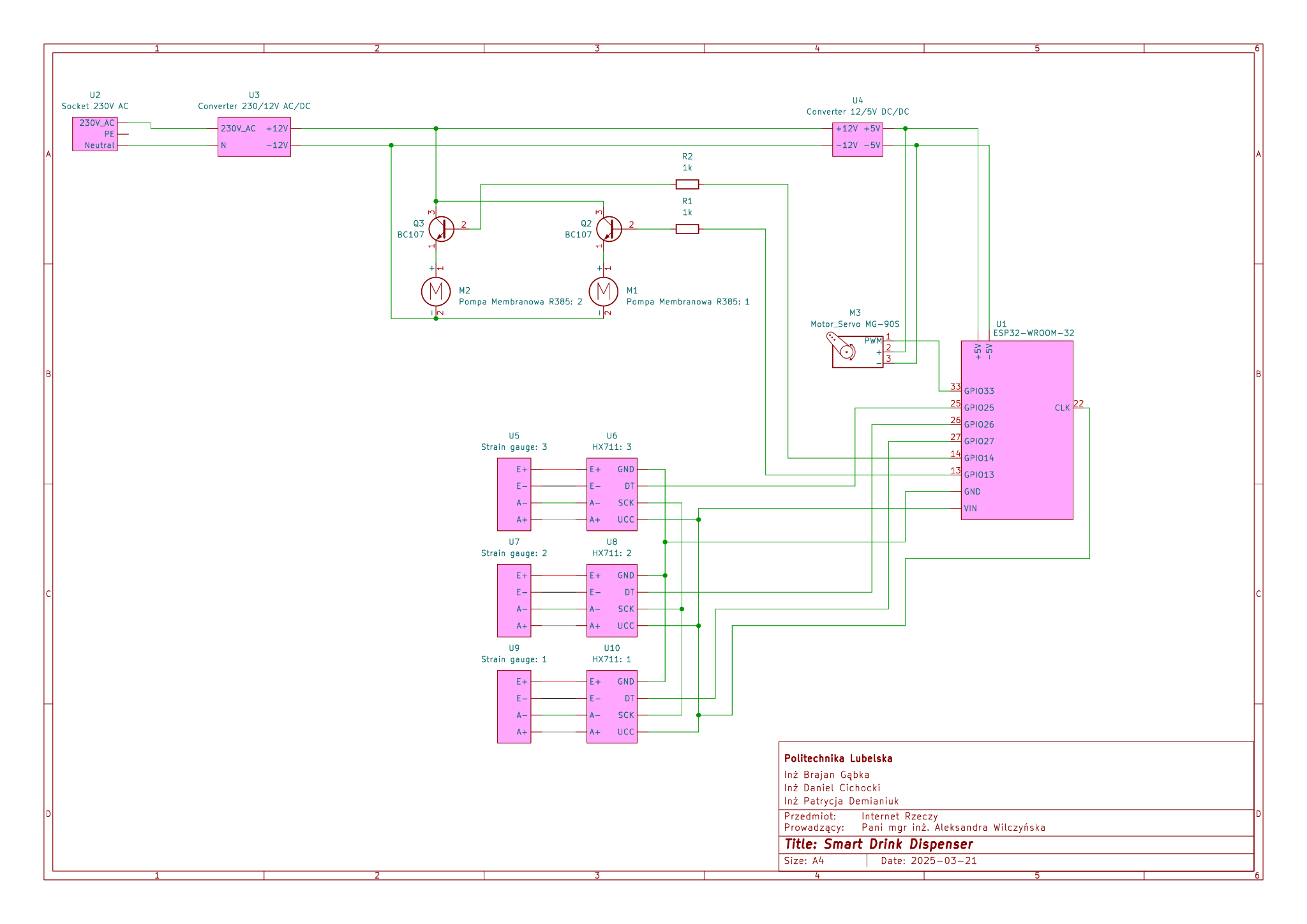
Demianiuk Patrycja

Gąbka Brajan

Politechnika Lubelska

Lublin, 2025 r.

1. **Schemat połączeń elektrycznych**



*Rys. 1. Schemat połączeń elektrycznych.*

1. **Opracowany fragment kodu**

| 00001 #include "HX711.h"  00002 #include <ESP32Servo.h>  00003  00004 #define pumpPin1 13  00005 #define pumpPin2 14  00006  00007 #define CLK 22  00008 #define dataOUT1 27  00009 #define dataOUT2 26  00010 #define dataOUT3 25  00011 #define servoPin 33  00012  00013 #define calibration\_factor1 -3350.0  00014 #define calibration\_factor2 -3350.0  00015 #define calibration\_factor3 -3350.0  00016  00017 #define empty\_glass 200.0  00018 #define full\_glass 450.0  00019  00020  00021 HX711 scale1;  00022 HX711 scale2;  00023 HX711 scale3;  00024 Servo servo;  00025  00032 class Glass {  00033 public:  00034 static float amount\_1;  00035 static float amount\_2;  00036 int position\_glass;  00037 static int position\_servo;  00038  00039 Glass(int pos) {  00040 this->position\_glass = pos;  00041 }  00042  00048 void pour () {  00049 moveServomechanism();  00050 float weight = returnWeight();  00051 float weight\_empty\_glass\_and\_amount\_1;  00052 if (weight < empty\_glass -0) {  00053 // brak szklanki  00054 return;  00055 }  00056 while (true) {  00057 weight = returnWeight();  00058 if (weight < empty\_glass + Glass::amount\_1) {  00059 digitalWrite(pumpPin1, HIGH);  00060 } else {  00061 digitalWrite(pumpPin1, LOW);  00062 weight\_empty\_glass\_and\_amount\_1 = weight;  00063 break;  00064 }  00065 }  00066 while (true) {  00067 weight = returnWeight();  00068 if (weight < weight\_empty\_glass\_and\_amount\_1 + Glass::amount\_2) {  00069 digitalWrite(pumpPin2, HIGH);  00070 } else {  00071 digitalWrite(pumpPin2, LOW);  00072 break;  00073 }  00074 }  00075 }  00076  00082 static float returnWeight() {  00083 switch(Glass::position\_servo) {  00084 case 0:  00085 return -1.f;  00086 case 1:  00087 return scale1.get\_units();  00088 case 2:  00089 return scale2.get\_units();  00090 case 3:  00091 return scale3.get\_units();  00092 }  00093 }  00094  00103 bool moveServomechanism(bool default\_position = false) {  00104 if (default\_position) {  00105 servo.write(0);  00106 Glass::position\_servo = 0;  00107 return true;  00108  00109 } else if (this->position\_glass == 1) {  00110 servo.write(45);  00111  00112 } else if (this->position\_glass == 2) {  00113 servo.write(90);  00114  00115 } else if (this->position\_glass == 3) {  00116 servo.write(135);  00117  00118 }  00119 Glass::position\_servo = this->position\_glass;  00120 return false;  00121 }  00122  00126 void default\_position() {  00127 digitalWrite(pumpPin1, LOW);  00128 digitalWrite(pumpPin2, LOW);  00129 moveServomechanism(true);  00130 }  00131  00138 static bool change\_recipt(float amount\_1, float amount\_2) {  00139 Glass::amount\_1 = amount\_1;  00140 Glass::amount\_2 = amount\_2;  00141 return true;  00142 }  00143 };  00144  00145 float Glass::amount\_1 = 0.0f;  00146 float Glass::amount\_2 = 0.0f;  00147  00148 void setup() {  00149 pinMode(pumpPin1, OUTPUT);  00150 pinMode(pumpPin2, OUTPUT);  00151 digitalWrite(pumpPin1, LOW);  00152 digitalWrite(pumpPin2, LOW);  00153 scale1.begin(dataOUT1, CLK);  00154 scale2.begin(dataOUT2, CLK);  00155 scale3.begin(dataOUT3, CLK);  00156 servo.attach(servoPin);  00157 Serial.begin(9600);  00158 Serial.println("Start programu");  00159 scale1.set\_scale(calibration\_factor1);  00160 scale2.set\_scale(calibration\_factor2);  00161 scale3.set\_scale(calibration\_factor3);  00162 scale1.tare();  00163 scale2.tare();  00164 scale3.tare();  00165  00166 Glass::change\_recipt(50.0f, 60.0f);  00167 Glass glass1(1); // obiekt szklana1  00168 Glass glass2(2); // obiekt szklana2  00169 Glass glass3(3); // obiekt szklana3  00170 }  00171  00172 void loop() {  00173  00174  00175 } |
| --- |

Repozytorium GitHub: <https://github.com/Duke-Axer/Smart-Drink-Dispenser>

**Dokumentacja klasy Glass**

Klasa Glass odpowiada za obsługę pojedynczej szklanki. Umożliwia kontrolowanie ruchów serwomechanizmu w celu ustawienia kranu w odpowiedniej pozycji, monitorowanie wagi szklanki i sterowanie pracą pomp do nalania odpowiedniej ilości płynów. Definicja w linii **32** pliku **Smart\_Drink\_Dispenser.ino**.

**Metody publiczne**

* **Glass** (int pos)
* void **pour** ()

*Główna funkcja odpowiedzialna za nalewanie cieczy do szklanek.*

* bool **moveServomechanism** (bool **default\_position**=false)

*Funkcja obsługująca sterowanie serwomechanizmem.*

* void **default\_position** ()

*Funkcja ustawiająca serwomechanizm i pompy do pozycji domyślnej.*

### **Statyczne metody publiczne**

* static float **returnWeight** ()

*Sprawdza wagę na pozycji, na której obecnie znajduje się serwomechanizm.*

* static bool **change\_recipt** (float **amount\_1**, float **amount\_2**)

*Modyfikuje przepis na napój poprzez zmianę ilości płynów, które mają zostać nalane.*

### **Atrybuty publiczne**

* int **position\_glass**

*Pozycja szklanki.*

### **Statyczne atrybuty publiczne**

* static float **amount\_1** = 0.0f

*Porcja napoju nr 1.*

* static float **amount\_2** = 0.0f

*Porcja napoju nr 2.*

* static int **position\_servo**

*Pozycja serwomechanizmu.*

### **Dokumentacja konstruktora i destruktora**

#### Glass::Glass (int pos)[inline]

Definicja w linii **39** pliku **Smart\_Drink\_Dispenser.ino**.

### **Dokumentacja funkcji składowych**

#### static bool Glass::change\_recipt (float amount\_1, float amount\_2)[inline], [static]

Modyfikuje przepis. Definicja w linii **138** pliku **Smart\_Drink\_Dispenser.ino**.

##### **Parametry**

| *amount\_1* | Porcja pierwszego napoju w przepisie. |
| --- | --- |
| *amount\_2* | Porcja drugiego napoju w przepisie. |

#### void Glass::default\_position ()[inline]

Funkcja ustawiająca serwomechanizm i pompy do pozycji domyślnej.

Definicja w linii **126** pliku **Smart\_Drink\_Dispenser.ino**.

#### bool Glass::moveServomechanism (bool default\_position = false)[inline]

Funkcja obsługująca sterowanie serwomechanizmem.

Serwomechanizm ustawia się na pozycji, na której jest szklanka. Po ostatniej pozycji wraca do domyślnej. Definicja w linii **103** pliku **Smart\_Drink\_Dispenser.ino**.

##### **Parametry**

| *default\_position* | Czy serwomechanizm ma wrócić do pozycji domyślnej? |
| --- | --- |

##### **Zwraca**

Jeśli serwomechanizm znajduje się na domyślnej pozycji zwraca TRUE, w przeciwnym razie zwraca FALSE.

#### void Glass::pour ()[inline]

Główna funkcja odpowiedzialna za nalewanie do szklanek.

Ustawia serwomechanizm i nalewa odpowiednią porcję obu napojów do szklanki. Definicja w linii **48** pliku **Smart\_Drink\_Dispenser.ino**.

#### static float Glass::returnWeight ()[inline], [static]

Sprawdza wagę na pozycji, na której jest teraz serwomechanizm. Definicja w linii **82** pliku **Smart\_Drink\_Dispenser.ino**.

**Zwraca**

Wartość float. Jeśli serwomechanizm znajduje się w pozycji domyślnej zwraca -1. Dla pozostałych pozycji zwraca zmierzoną masę.

### **Dokumentacja atrybutów składowych**

#### float Glass::amount\_1 = 0.0f[static]

Porcja napoju nr 1. Definicja w linii **34** pliku **Smart\_Drink\_Dispenser.ino**.

#### float Glass::amount\_2 = 0.0f[static]

Porcja napoju nr 2. Definicja w linii **35** pliku **Smart\_Drink\_Dispenser.ino**.

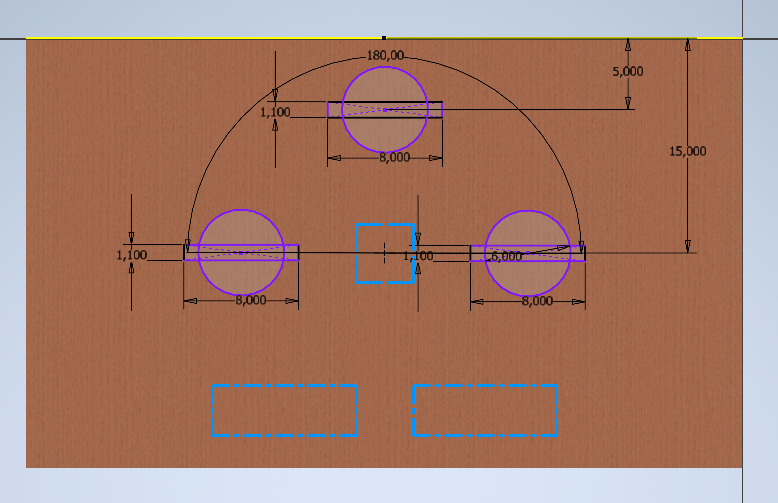
#### int Glass::position\_glass

Pozycja szklanki. Definicja w linii **36** pliku **Smart\_Drink\_Dispenser.ino**.

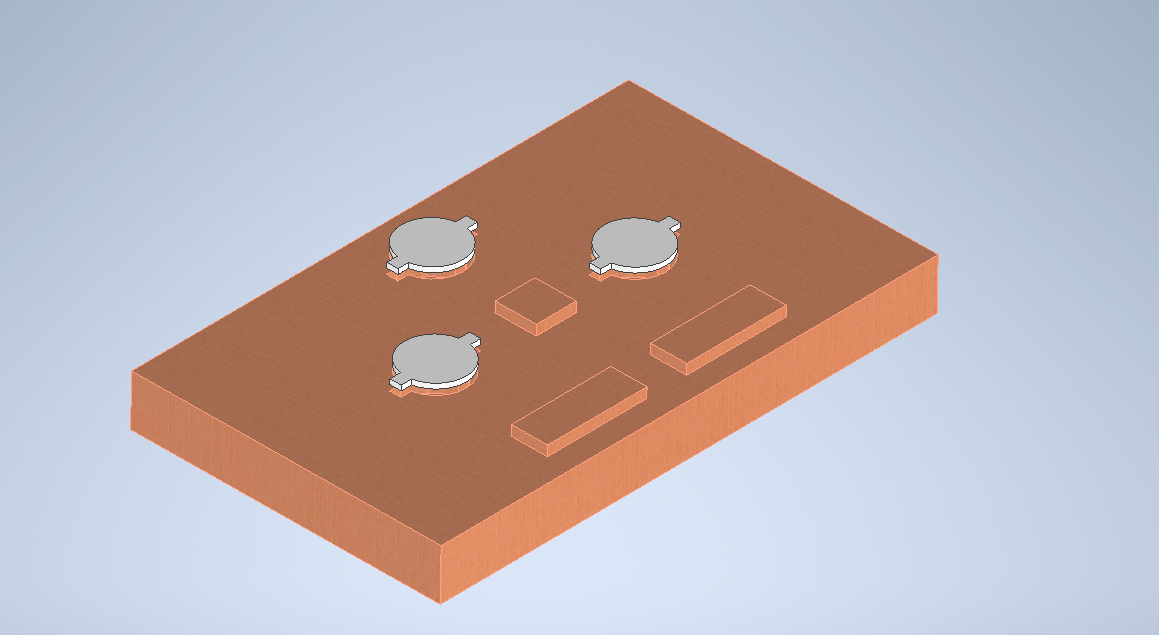
#### int Glass::position\_servo[static]

Pozycja serwomechanizmu. Definicja w linii **37** pliku **Smart\_Drink\_Dispenser.ino**.

1. **Projekt podstawy**



*Rys. 2. Szkic wraz z wymiarami.*



*Rys. 3. Wizualizacja.*