Filip-George MANOLE (78761) - Self Balance Robot

Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

- Scopul proiectului este realizarea unui robot pe doua roti care isi mentine echilibrul.
- Utilitatea acestui proiect se poate vedea, de exemplu, in modul in care sunt folosite si placile Hoverboard (de altfel, sursa inspiratiei mele): atat pentru transport, cat si pentru distractie.

Descriere generală

Descriere minimala

Se va citi regulat (la minim 100Hz) date de la senzorul de accelerometru + giroscop si se va face comensarea erorii prin actionarea motoarelor. Motoarele se actioneaza cu ajutorul driverelor.

Schema Bloc

Hardware Design

Lista de Piese

- Placa de baza
- 3x sursa coboratoare
- 2x motor Pololu 1573
- 2x driver motor VNH2SP30
- pereche roti Polulu 60 x 8 mm - roșii
- senzor BNO055
- baterie LiPo ZIPPY 7.4V 2200mA
- voltmetru

Sechema electrica

Scurta descriere

Intreg robotul este alimentat de la bateria de 7.4V. Pentru a verifica starea bateriei am legat un voltmetru la bornele ei.

Motoarele functioneaza la 6V si pot consuma un curent maxim de 6.5A. Pentru fiecate motor am folosit o sursa coboratoare pentru a regla voltajul la 6V si care pot da un curent de pana la 5A pentru fiecare motor.

Pentru a alimenta placa de baza la 5V am folosit alta sursa coboratoare care da un curent mai mic.

Software Design

Mediul de dezvoltare folosit este Atmel Studio.

Librarii

- I2C master + MPU6050
- I2C + BNO055

Implementare:

- Controlul motoarelor se face cu ajutorul driverelor. Driverul primeste un semnal PWM pentru a stabili viteza motoarelor, si 2 biti care in functie de cum ii setez stabileste directia sau opreste motorul.

```
/* Initializare PWM */
void motor_init(void);

void move_forward(uint8_t ocr);

void move_backward(uint8_t ocr);

void move_stop();
```

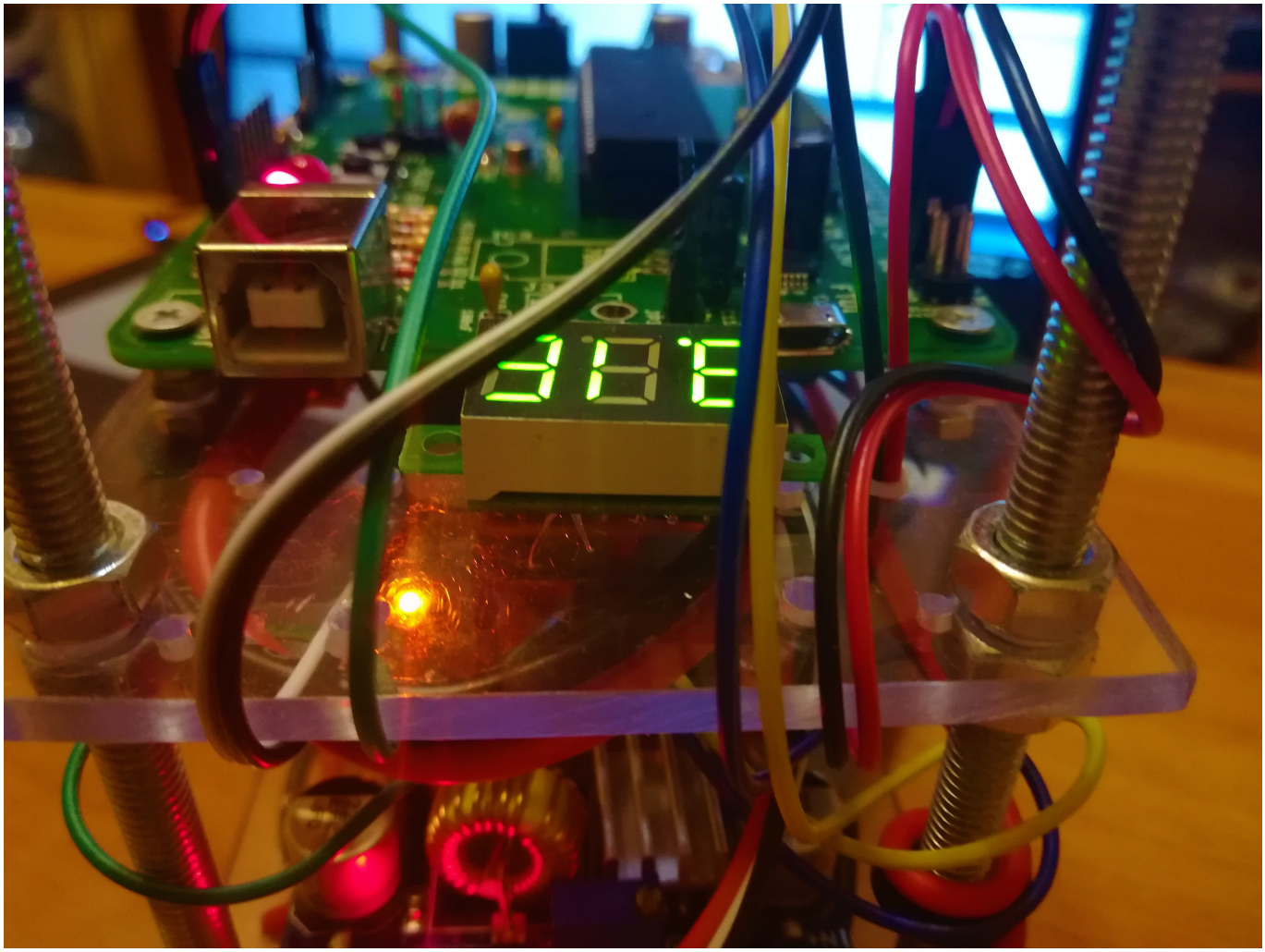
- Unghiul de inclinare al robotului l-am luat cu ajutorul bibliotecii pentru BNO055. Citesc datele senzorului, si ma intereseaza doar valoarea de pe axa z din unghiurile Euler.
- Controlerul PID este executat odata la 100Hz cu ajutorul unei intreruperi date de timerul 1. Valoare input este unghiul citit cu ajutorul accelerometului. Valoarea rezultata in urma calcularii comensarii erorii este folosita pentru a stabili directia si viteza motoarelor.

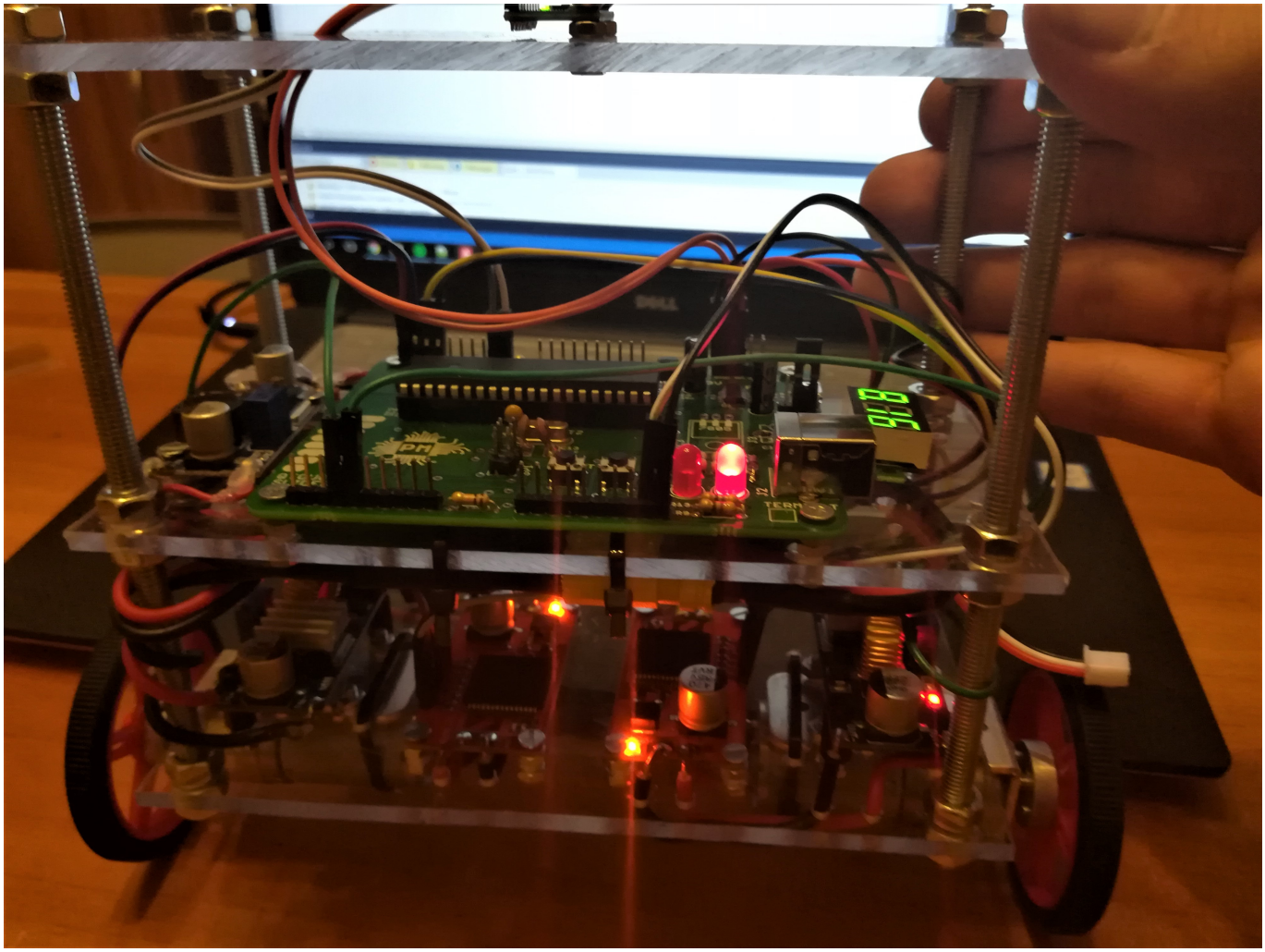
```
double compute(int input)
{

        static int lastErr = 0;
        static int errSum = 0;

        int error = SETPOINT - input;
        errSum += error;
        int dErr = (error - lastErr);
```

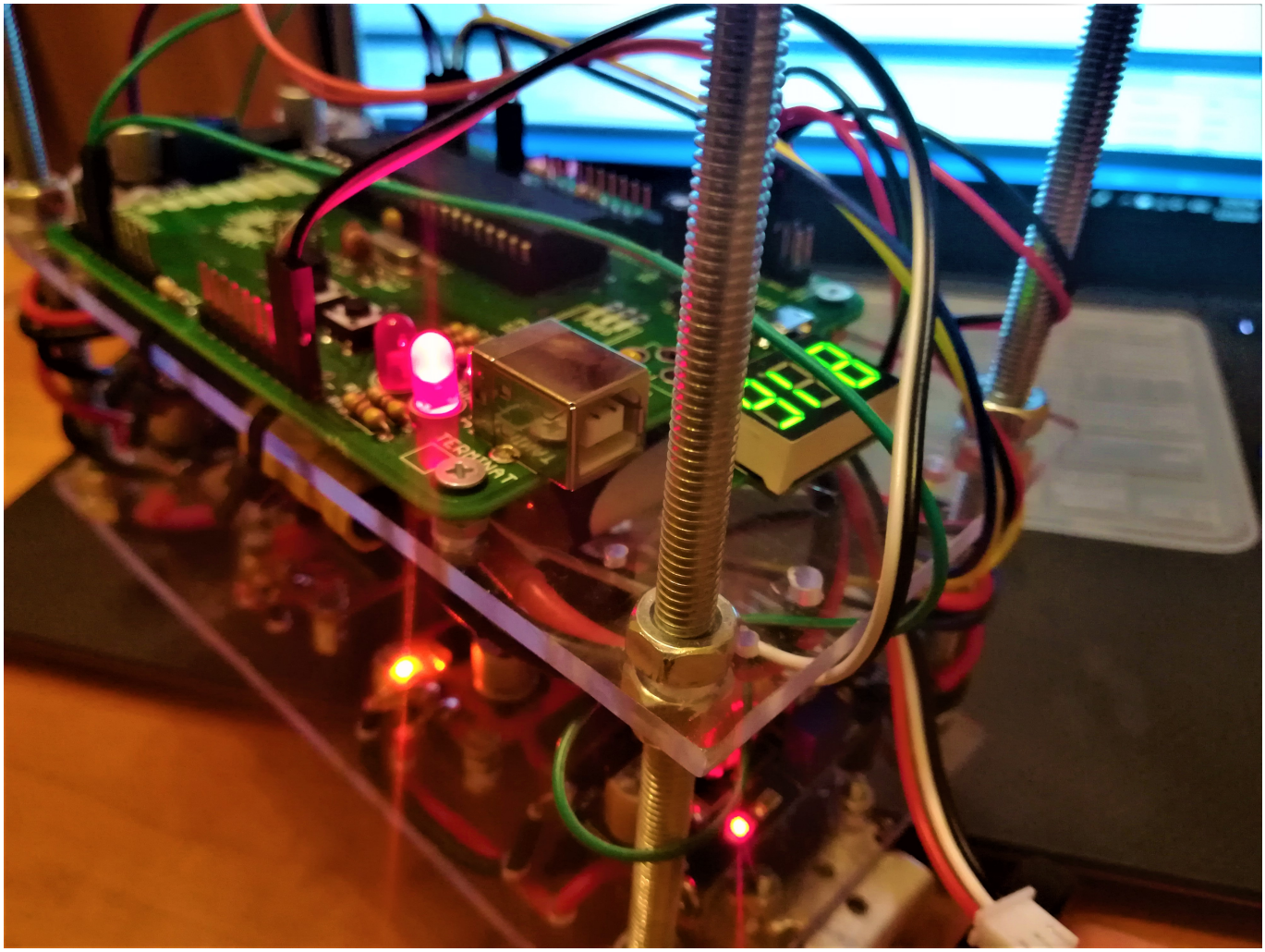
```
        /*Remember some variables for next time*/
        lastErr = error;

        /*Compute PID Output*/
        return Kp * error + Ki * errSum * SAMPLE_TIME + Kd * dErr /
SAMPLE_TIME;

}
```

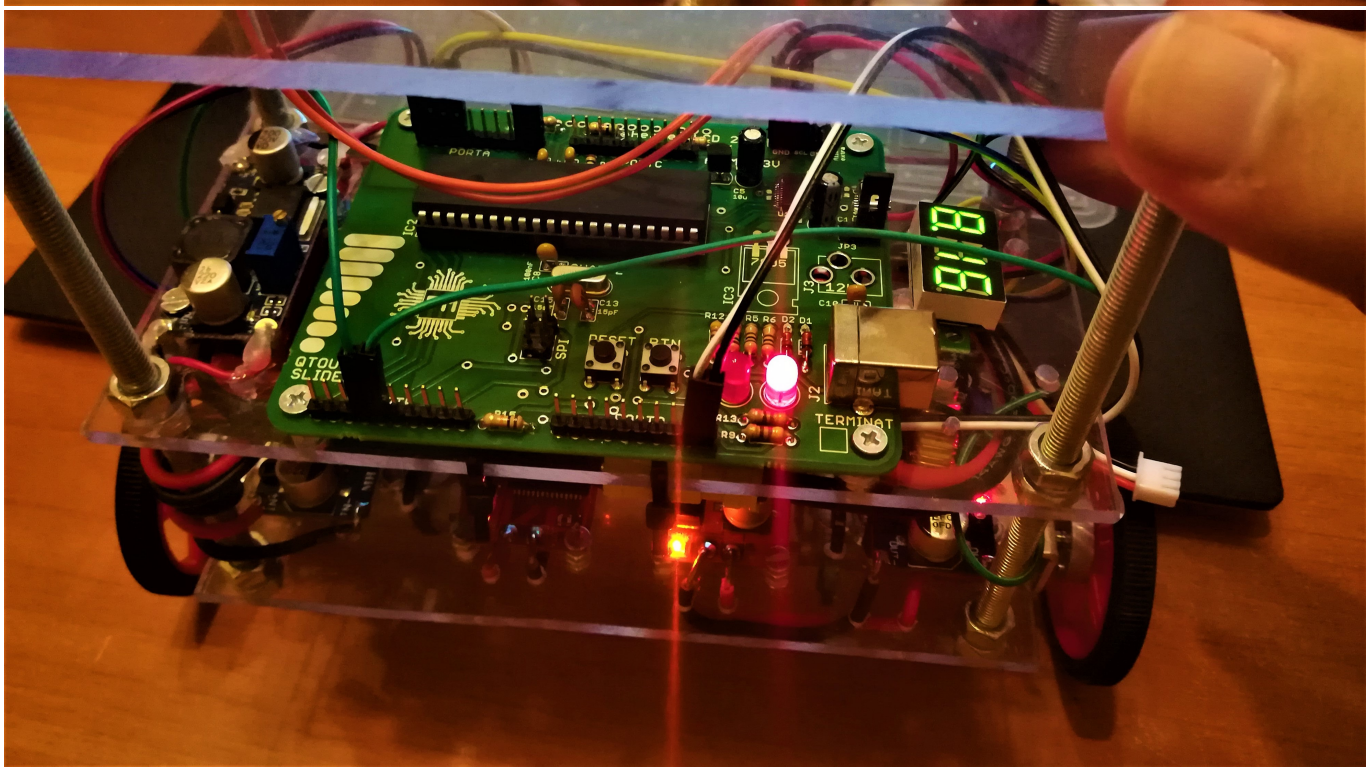
- Pentru debug am folosit un modul pentru seriala FT232.

Rezultate Obţinute









Software

GitHub - implementarea software a proiectului

Jurnal

- [12 Mai 2018] Realizarea hardware a proiectului (asamblare robot, prindere piese si legaturile electrice).
- [19 Mai 2018] Controlul motoarelor din PWM, citire date de la MPU6050 si implementare PID.
- [22 Mai 2018] Testare PID, reglari.
- [24 Mai 2018] Schimbare senzor MPU6050 cu BNO055 si reglare PID.

Bibliografie/Resurse

Specificatii componente hardware.

- BNO055
- MPU6050
- Motor
- Driver

Biblioteci/tutoriale folosite

- Tutorial MPU6050 si biblioteca folosita
- Proiectul de unde m-am inspirat + biblioteca senzor BNO055: TWIP
- Documentația în format PDF