# Stabilizácia robota pomocou gyroskopu

Michal CHOVANEC

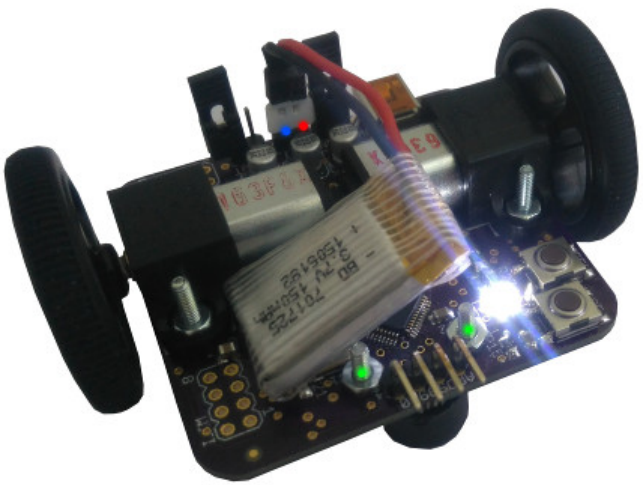
*September 2017*

# Cieľ

Naučiť študentov minimálne základy riadania v reálnom čas (odozva jednotky ms) v C++.
Robot bude stabilizovaný pomocou gyroskopu, bude schopný ísť rovno a bude kompenzovať pôsobiace bočné sily.

- Modul gyroskopu
- Základné triedy a UML
- Implementácia v C++ pomocou funktorov

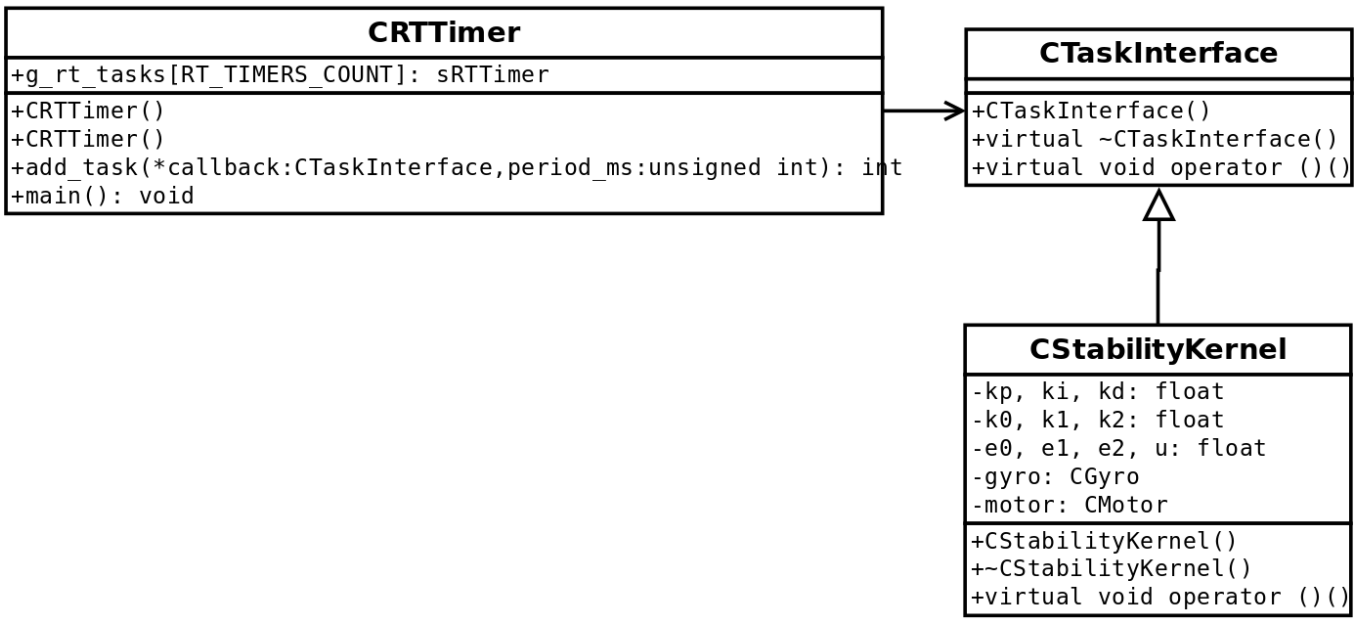
# Modul gyroskopu



- gyroskop LSM6DS0, i2c, 6DOF
- Zvolená konfigurácia

$$T_s = 10ms \quad (1ms)$$
$$scale = 500^\circ/s \quad (2000^\circ/s)$$
$$resolution = 17.50m^\circ/bit \quad (8.75m^\circ/bit)$$

# UML diagram




- **CRTTimer** : trieda pre spúšťanie úloh v reálnom čase
- **CTaskInterface** : virtuálna trieda, predok pre všetky úlohy bežiace v CRTTimer
- **CStabilityKernel** : potomok CTaskInterface, obsahuje hlavný riadiaci blok na stabilizáciu robota

# Diskrétny tvar PID

$$u(n) = u(n-1) + k_0 e(n) + k_1 e(n-1) + k_2 e(n-2)$$
$$k_0 = k_p + k_i + k_d$$
$$k_1 = -k_p - 2k_d$$
$$k_2 = k_d$$

- Ľahko sa implementuje programovo
- Odpadajú "Arduino PID" problémy napr : $err\_sum+ = e(n)$, $dif = e(n) - e(n-1)$
- Jednoduchý antiwindup : obmedzenie rozsahu u(n)
- Konštanty $k_p, k_i, k_d$ určené experimentálne, pre túto sústavu pochopiteľne $k_i = 0$

# Implementácia v C++

```
1   void CStabilityKernel::operator()()
2   {
3       float speed = 0; //change for forward motion
4
5       gyro.read();
6
7       float angle = 0.0;
8
9       //transform axis (resolution and orientation),
10      //for nice PID constants
11      angle = -gyro.angles.y*0.1;
12
13      e2 = e1;
14      e1 = e0;
15      //subtract required and meassured value
16      e0 = 0.0 - angle;
17
18      //process PID controller
19      u+= k0*e0 + k1*e1 + k2*e2;
20
21      //output for motors, limit output values
22      int left  = saturate(u + speed, -256, 256);
23      int right = saturate(-u + speed, -256, 256);
24
25      motor.run(left, right);
26  }
```

# Ďakujem za pozornosť

https://github.com/michalnand/y_robot_course

michal.nand@gmail.com