ECE MBD: Laboratory session 7

Balanbot – Interface setup for data acquisition in external mode and Rapid Control Prototyping (RCP)

During this session you shall create the interfaces in the Simulink environment for the Balanbot and deploy the model developed in LAB6 at the target (Arduino Mega) in External Mode by means of Automatic Code Generation. The Arduino Mega is connected to an MPU6050 which is used as a sensor and motor control board. For the operation of this board first some registers have to be configured.

Laboratorywork

Model Preparation

1. Prepare a new model in your MATLAB working directory and configure it as follows:
   - Select the proper target by setting up your model in the Configuration Parameters Panel/Hardware Implementation to the Arduino Mega 2560.
     The detection of the COM should work automatically; but in case that it does not, then change the Host-board connection mode to Manual and then specify the COM port of your serial communication.
   - Set the fixed-step solver of your model to 10ms (NF1 of previous lab).

2. The Simulink Model shall have the following structure:

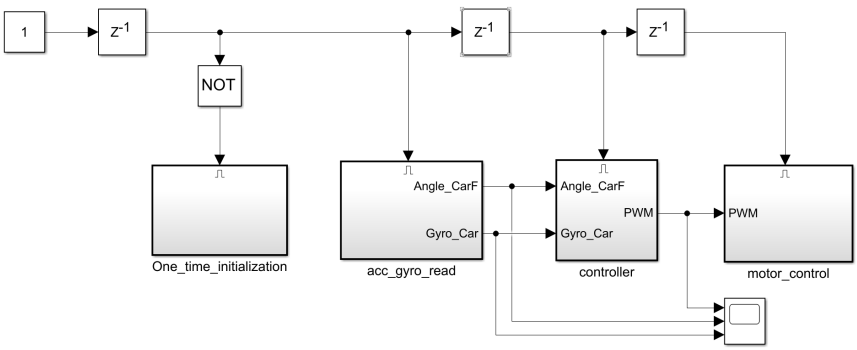

Figure 1: Model framework

The enabled subsystems represent a consecutive execution of the implemented functions. The first subsystem is the initialization of the sensors and outputs and it is executed only once when the model is launched.

System Initialization (MPU6050)

3. In order to operate the MPU6050 it has to be configured first, therefore the registers of the board have to be set using the I2C and Digital output blocks from the Simulink Arduino library.
    - Implement the initialization sequence as shown below. An example of how to configure the I2C block is shown in Figure 3. The corresponding slave device address is 0x68

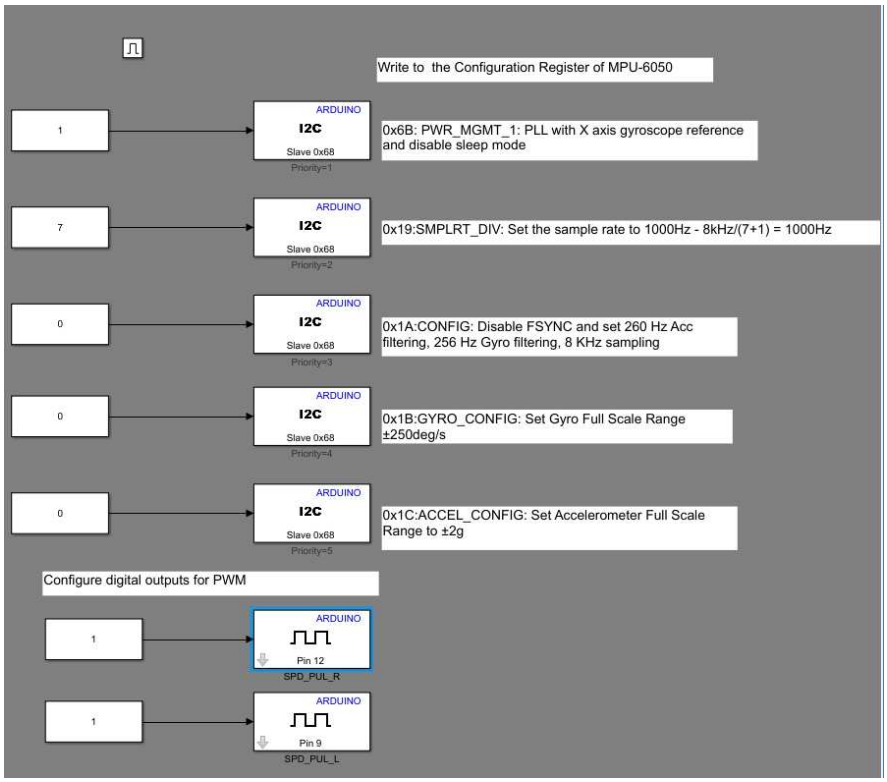

Figure 2: One time initialization

A comprehensive list of the MPU 6050 and the registers which are configured can be found at: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

Here it is the example of the first block:

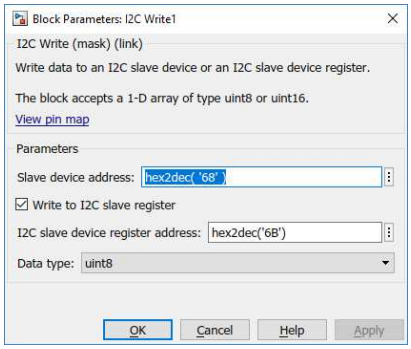

Figure 3: I2C Block Configuration

Note: For your information, the model above implements the following C code:

```
  i2cData[0] = 7; // Set the sample rate to 1000Hz - 8kHz/(7+1) = 1000Hz
  i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc filtering, 256 Hz
Gyro filtering, 8 KHz sampling
  i2cData[2] = 0x00; // Set Gyro Full Scale Range to ±250deg/s
  i2cData[3] = 0x00; // Set Accelerometer Full Scale Range to ±2g

  while (i2cWrite(0x19, i2cData, 4, false)); // Write to all four registers
at once
  while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis gyroscope reference
and disable sleep mode
```

Read Gyro Sensor Data

4. The second subsystem shall read out the sensor values from the target at a 10ms rate and compute the angular speeds and accelerations; therefore:
   - Use the I2C blocks to read out the sensor data (as shown below)
   - The information from the I2C has to be converted into usable data. Implement the two calculation (once for the angle and accelerator) blocks as MATLAB functions.

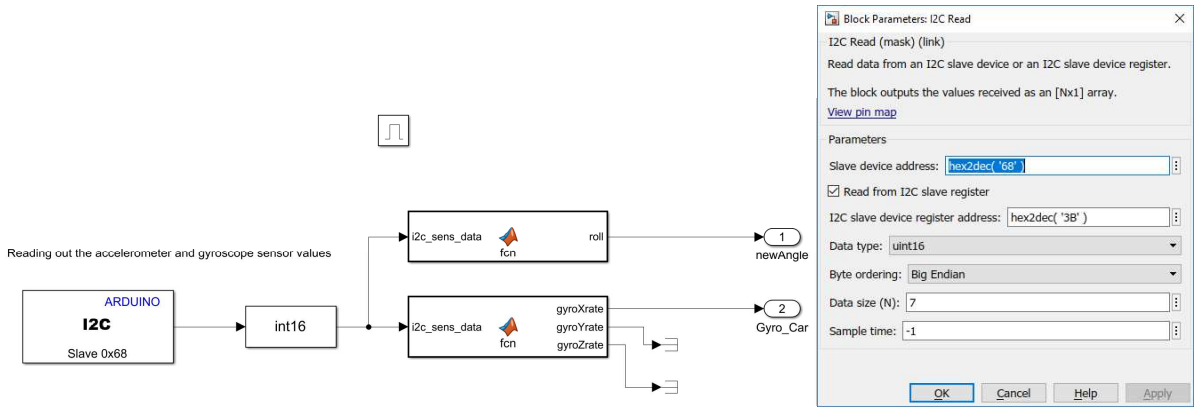

Figure 4: I2C Sensor Configuration

Function(s) for gathering the angle and acceleration:

```
function roll = fcn(i2c_sens_data)
accX = double(i2c_sens_data(1));
accY = double(i2c_sens_data(2));
accZ = double(i2c_sens_data(3));
roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * 180/pi;
```

```
function [gyroXrate,gyroYrate,gyroZrate] = fcn(i2c_sens_data)
gyroX = double(i2c_sens_data(5));
gyroY = double(i2c_sens_data(6));
gyroZ = double(i2c_sens_data(7));
gyroXrate = gyroX/131;
gyroYrate = gyroY/131;
gyroZrate = gyroZ/131;
```

5. Test the implemented functions by running the model in external mode. Observe the signals provided by the sensor using a scope. Make a comparison between the raw values and the filtered angle, report your findings.

Actuators (motors)

6. The fourth subsystem will write the digital signals that control the motors. Use the digital write blocks from the Simulink Arduino library to implement the motor control phase.
    - The speed of the motor is configured via PWM blocks from the library (max. 255).
        i. In order to test the algorithm, integrate a slider gain for the PWM as it allows you to change the speed values of the motors in external mode.
    - The rotational direction of the motor is configured via digital pins see Figure 5.
        i. Use a manual switch in the Simulink model to switch the motor drive direction during external mode simulation.

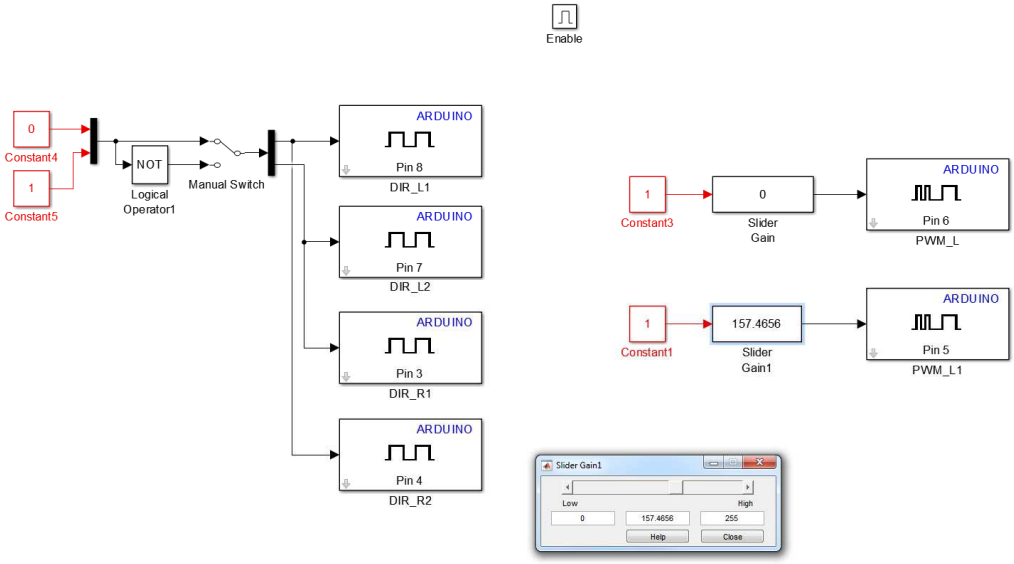

Figure 5: Motor control

7. Test your interfaces in External Mode together with the Balanbot
8. For the final setup, change the motor_control block as follows:

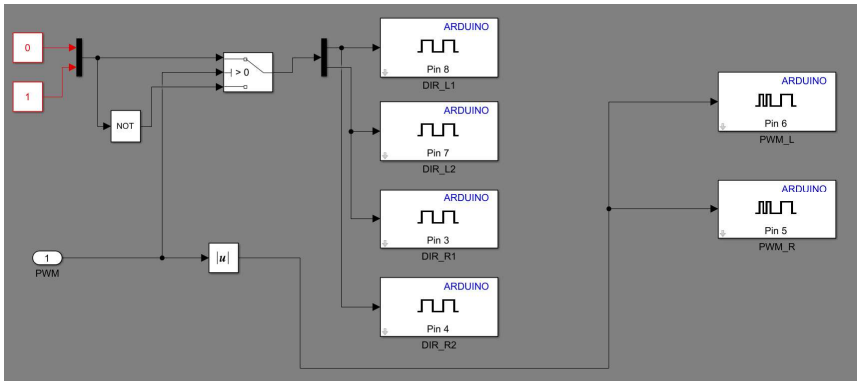

Figure 6: Final motor control

Controller

9. Integrate the third main block (controller) developed in Lab6.

   Remember NF3:
   - The proportional, integral and derivative gains (Kp, Ki and Kd; respectively) shall be adjustable during run
   - Kp shall be multiplied by angle (output of Kalman filter)
   - Ki shall be multiplied by the integral of angle (output of Kalman filter)
   - Kd shall be multiplied directly by newRate (input of the Kalman filter)
     - Note: In the hardware implementation this is a reliable signal coming from the gyroscope.
   - The output of the controller (sum of the three previous parts) shall be saturated to -255 and 255
10. Integrate a manual switch to the output of the controller which allows to stop the motor operation
11. Tune the controller for achieving a stable balanbot
    - Report your findings

From the exercises, it is expected that you report
- your development process of your solution: design and testing
- Add a short video of the working Balanbot

Organization:

The Balanbots are coordinated by Mr. Läßer and **have to** be returned to him as there is a limited number of robots available!!!

In order to finish your laboratory task please contact him for accessing the hardware.

Important:

Consider to use always the same Balanbot for your test, another Balanbot may react different as the hardware may have a different motor reaction, gear plays etc.