March 11, 2015

Aerospace System Guidance and Control
Lesson V

Arduino Basics

POLITECNICO DI MILANO

DIPARTIMENTO DI
INGEGNERIA AEROSPAZIALE

Skyward Experimental Rocketry

Politecnico di Milano

Author: Francescodario Cuzzocrea

Editor: Edoardo Codispoti

Abstract
In this lesson we will talk about the interface between the Arduino Board and Simulink

Website:
http://www.skywarder.eu

E-mail:
francescodario.cuzzocrea@skywarder.eu

Contents

Chapter 1

What is Arduino

Arduino is a single-board microcontroller inteded to make the application of interactive objects or enviroment more accessible. So Arduino can be used for fast prototiping, or for building small robots (or an airship, like in our case).
Pre-programmed into the on-board microcontroller chip is a boot loader that allows uploading programs into the microcontroller memory without needing a chip (device) programmer, simply by using the USB port. Arduino is provided with input/output functionality so the board can send/receive data to the external sensors. The behaviour of the board is managed by microcontroller based on the decisions implemented on the program that's running on the board. The board may also interact with the external enviroment by using actuator driven by the pro- gram trought the output channels.

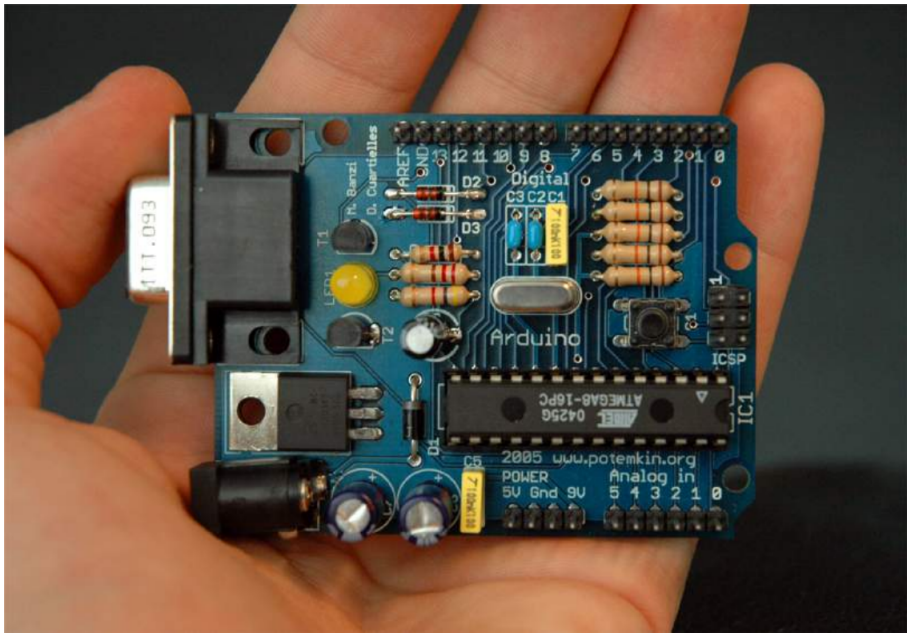

Figure 1.1: Arduino

Chapter 2

Programming Arduino with Simulink

Simulink can be enhanced with different kind of libraries, so we can simulate the behaviour of a large variety of system and plus, Simulink can be used to run our simulation on a targetted hardware board, in this case for example we are going to use an Arduino Uno, but also STM32 BeagleBoard and Raspberry Pi can be used, MathWorks is putting a lot of effort in supporting various kind of board at every new relase of MATLAB.

2.1 A first simple example

So let's start to play together with the Arduino Uno board and Simulink. In this first example you will learn how to interface Arduino with Simulink, so we can generate a firmware compatible with the board starting from our model built and tested on Simulink.

First open up Simulink, simply by typing simulink from the MATLAB command prompt, so the Simulink library browser will show up.

For this first simple example we are going to use blocks from the Simulink blockset and from the Simulink Support Package for Arduino Hardware blockset. Let's start a new blank project by clicking on File > New > Model. We want to control the digital output of the Arduino Board in order to made a led blink at specified time interval, so first we need to select a Pulse Generator, simply by drag and dropping the Pulse Generator block in the model window. We need the Pulse Generator in order to generate the signal that we will sent to the digital output of the Arduino Uno board :

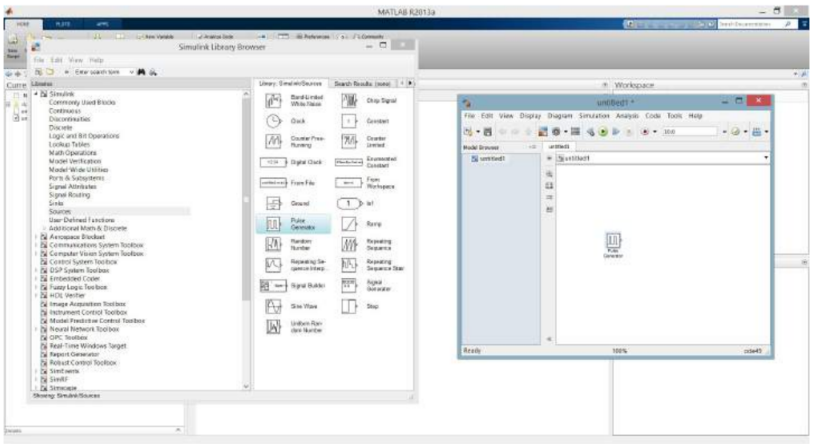

Figure 2.1: Pulse Generator

Then we need to connect the pulse generator to the digital outpu of the Arduino Uno board. So let's browse the Simulink Support Package for Arduino Hardware blockset and select the digital output block.

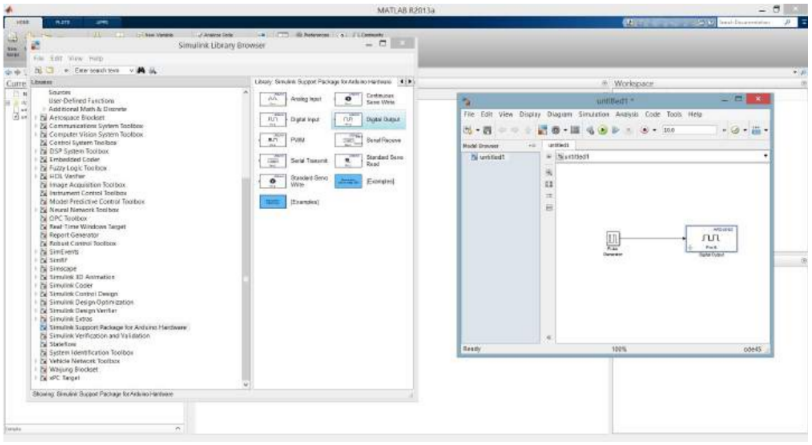

Figure 2.2: Digital Output Block

In order to made our model work, we need to set the corret pulse type of our pulse generator, so as our signal should control a digital output, we need to set the "sample based" pulse type and set the sample time to 0.1 (or wichever value you want to set) :

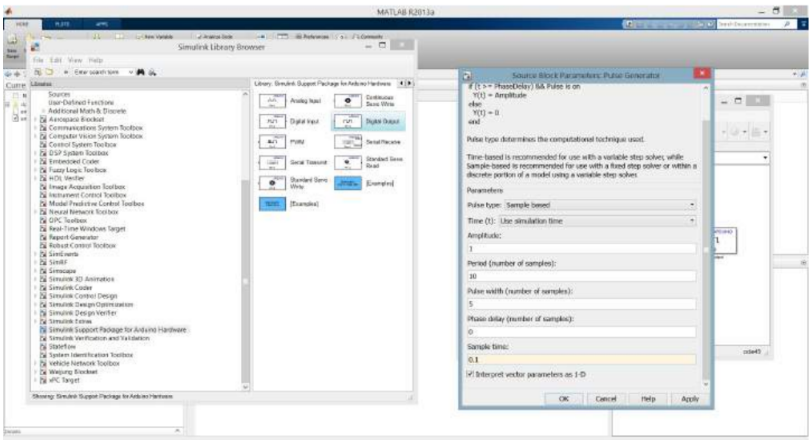

Figure 2.3: Sample Generator Options

Now we are ready to test our model consistency with Simulink. Let's connect a scope between the Pulse Generator block and the digital output block, and next cick Run. Double clicking on the scope will show us the results of our simulation :

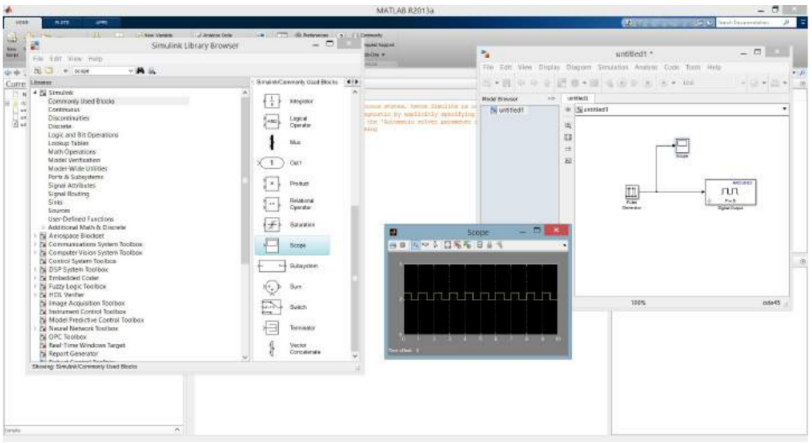

Figure 2.4: Simulation Results

Now we are ready to build the real circuit and to download the firmware to the Arduino Uno board.In the figure below is rapresented the wiring

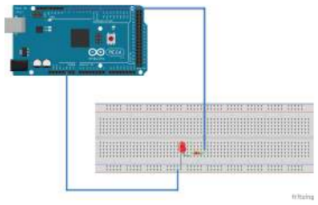

Figure 2.5: LED Wiring

So, after the wiring is done, we need to connect the Arduino Uno board to our PC trought the USB cable. On Simulink let's click on Tools > Run on Target Hardware > Prepare to Run. A window will pop-up, we need to chose the Arduino Uno target hardware, then click Apply. The Simulink defaults settings are perfectly suitable for our needs.

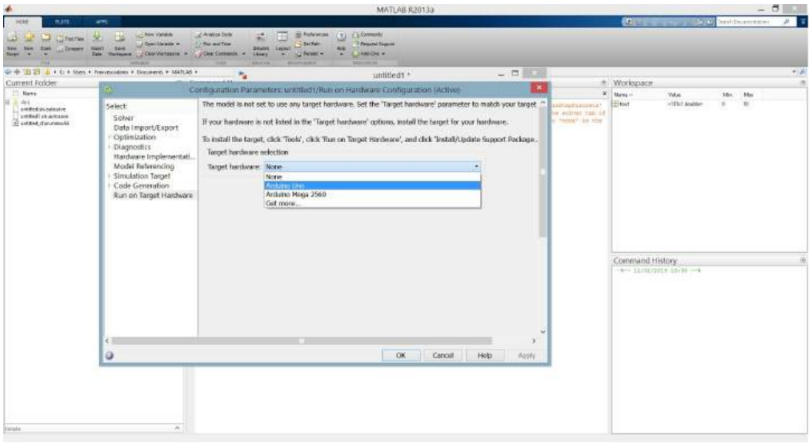

Figure 2.6: LED Wiring

Now click again on Tools > Run on Target Hardware > Run. Simulink will automatically generate the necessay C code based on our model, and will flash the firmware on the board. Take a look to the led, should blink.

2.2 Traffig Light Controller

In this example we will gonna use MathWorks StateFlow in order to simulate the logic of a Traffic Light, then we will download the model on Arduino. Stateflow is a graphical design and development tool for control and supervisory logic that can be used in conjuction with Simulink. So with Stateflow we can model and simulate combinatorial and sequential decision logic based on state machines and flow charts, like for example, a traffic light controller :

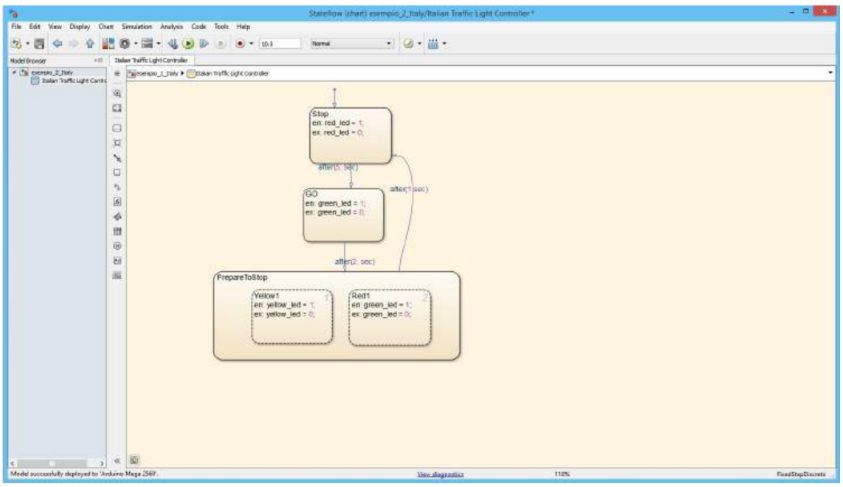

Figure 2.7: Stateflow

So, in order to use the stateflow chart, we need to connect the output of the state machine to Arduino Digital Output Block, build the circuit, and then we have to deploy the model on the hardware :

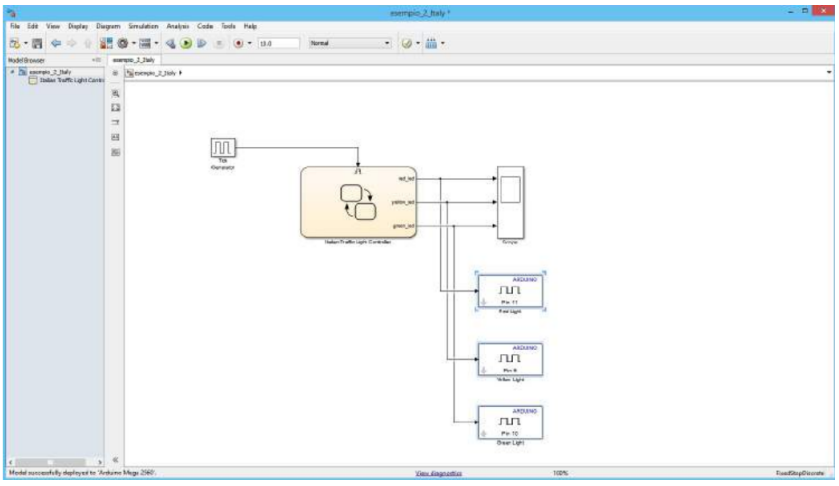

Figure 2.8: Traffic Light Controller

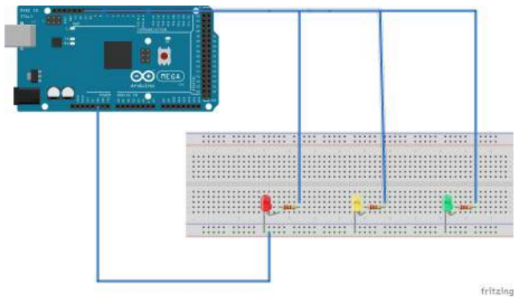

Figure 2.9: Traffic Light Controller Wiring

2.3 Servo Control

With the Simulink Support Package for Arduino we can also command a servo, trought the Standard Servo Write Block. Let's see an example. Open a new model, then input the desired shaft angle and connect the desired shaft angle block to the servo block. Then deploy the model on the hardware.

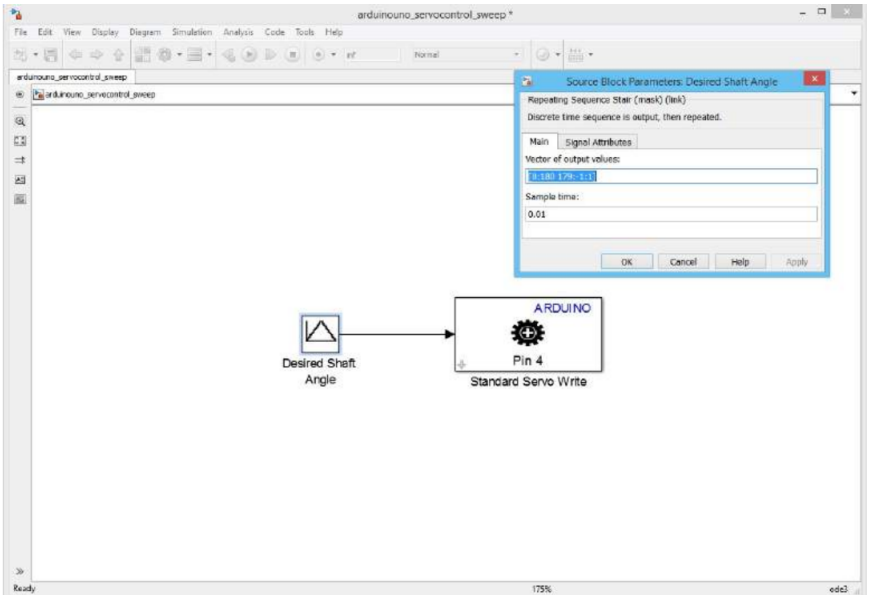

Figure 2.10: Servo Control

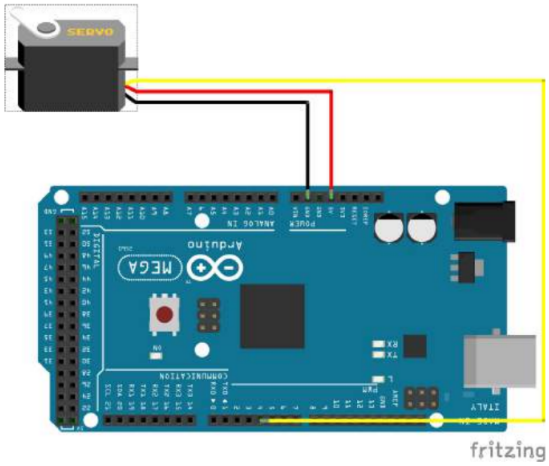

Figure 2.11: Servo Wiring

Chapter 3

Sensors Reading with Simulink and Arduino

3.1 What is a MEMs

The sensor that we will use are known as MEMs (Microelectromechanical systems). MEMS, is a technology that in its most general form can be defined as miniaturized mechanical and electro-mechanical elements (i.e., devices and structures) that are made using the techniques of microfabrication. The critical physical dimensions of MEMS devices can vary from well below one micron on the lower end of the dimensional spectrum, all the way to several millimeters. Likewise, the types of MEMS devices can vary from relatively simple structures having no moving elements, to extremely complex electromechanical systems with multiple moving elements under the control of integrated microelectronics.

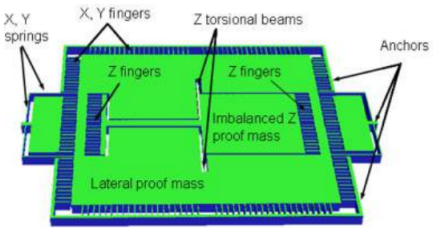

Figure 3.1: MEMs Accelerometer

MEMS typically converts a measured mechanical signal into an electrical signal The real potential of this kind of device, is that these miniatured sensors can me merged onto a common silicon substrate along with integrated circuit process sequences, and so you can interface a MEMs, for example, with an Arduino board .

3.2 MEMs communication interface

Our MEMs communicates with Arduino trought a bus called I2C (Inter-Integrated Circuit). The I2C bus transmits data and clock with SDA and SCL. . SCL is the clock line. It is used to synchronize all data transfers over the I2C bus. SDA is the data line. First thing to realize: SDA and SCL are open-drain (also known as open-collector in the TTL world), that is I2C master and slave devices can only drive these lines low or leave them open. The termination resistor (pull-up resistors) pulls the line up to Vcc if no I2C device is pulling it down. This allows for features like concurrent operation of more than one I2C master (if they are multi-master capable) or stretching (slaves can slow down communication by holding down SCL). The devices on the I2C bus are either masters or slaves. The master is always the device that drives the SCL clock line. The slaves are the devices that respond to the master. A slave cannot initiate a transfer over the I2C bus, only a master can do that. There can be, and usually are, multiple slaves on the I2C bus, however there is normally only one master. It is possible to have multiple masters, but it is unusual and not covered here. On your robot, the master will be your controller and the slaves will be our modules such as the SRF08 or CMPS03. Slaves will never initiate a transfer. Both master and slave can transfer data over the I2C bus, but that transfer is always controlled by the master. In our case the master is Arduino, the slave are the sensors. We can connect how many slaves that we want to one master trought the I2C interface (the sensors can share SDA and SCL lines). At this point should be clear the wiring : you simply have to connect SDA and SCL to the SDA and SCL pins of Arduino board, GND to GND and VCC to the sensors specified logic voltage (3.3V or 5V).

3.3 Skyward BLIMP Target

In order to make life easy to people who wants to drive sensors trough Simulink, we have developed a Simulink Library that contains all the device drivers blocks that we will going to use on the BLIMP.

The installation is pretty easy, you only ave to place the “Libreria” folder somewere, then you have to open the folder in matlab (the folder contents should be shown in MATLAB current folder) and run the SkywardConfigureScript. The script will take care of adding corrects path to MATLAB and to add the library to simulink.

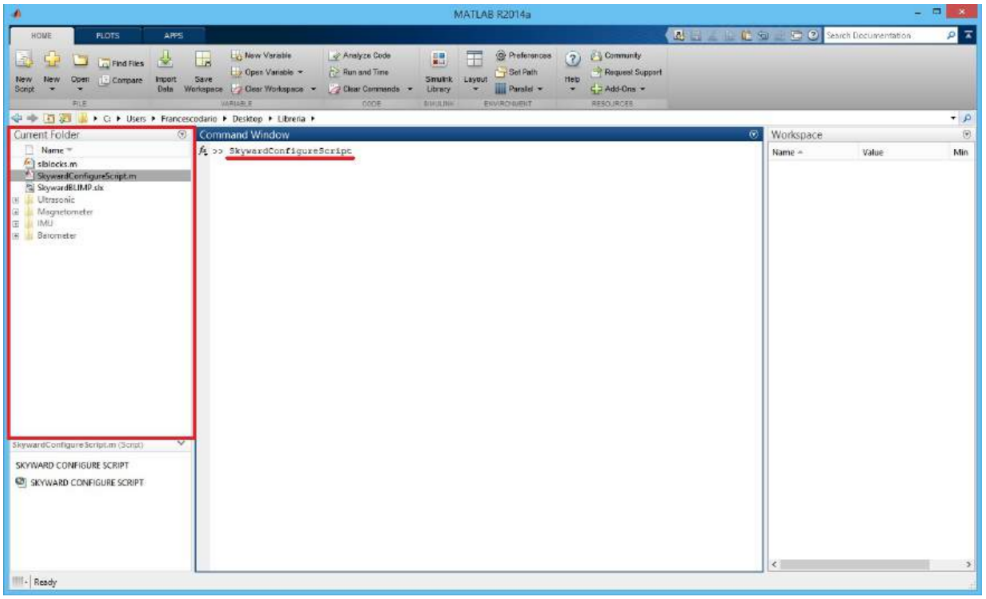

Figure 3.2: BLIMP Target Installation

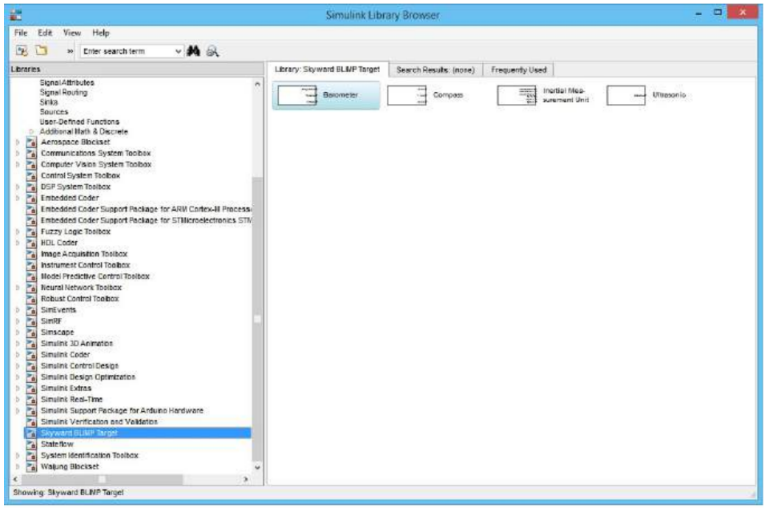

Figure 3.3: BLIMP Target Library

3.4 Read Data from Sensors

Once we have create a new model using those blocks, we can for example see what kinds of data are processed trought the Serial port by adding a Serial Transmit block and using the Serial Receive block. The serial receive can send those datas to workspace, or you can connect a scope, for example, to see the graph of data versus time.

Remember to always convert datas trought data type conversion block to uint8 right before the Serial Transmit block (because this block can only print integer variables).
This is an example of a model to create in order to send datas to the serial port :

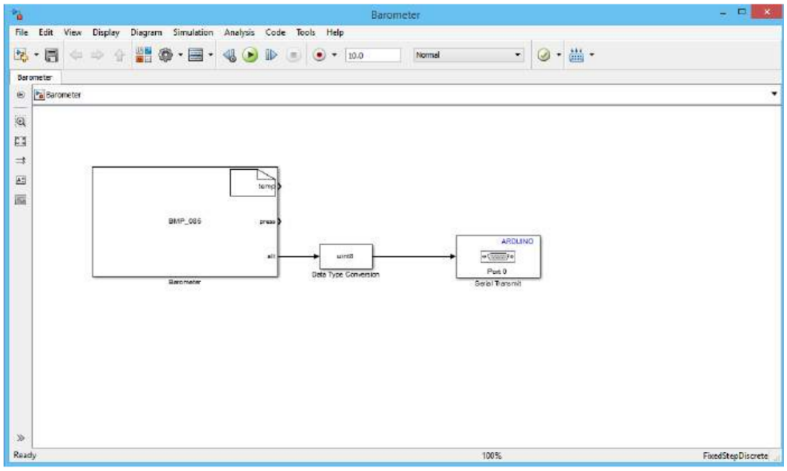

Figure 3.4: Serial Transmit

And his is the model to create for the serial receive :

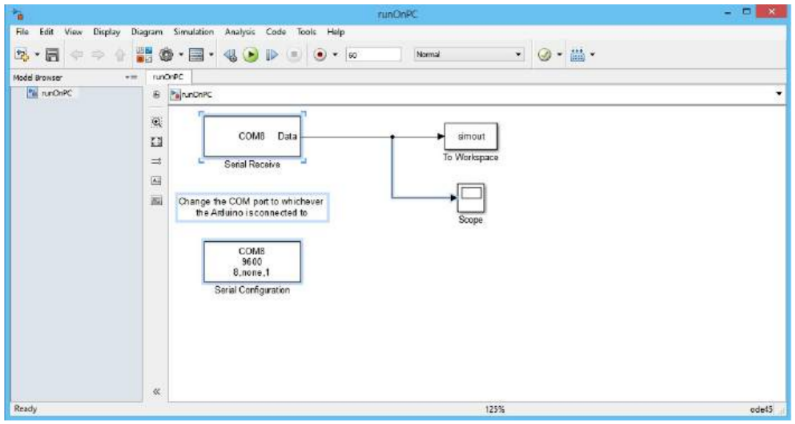

Figure 3.5: Serial Receive

Keep in mind that “On Target” execution is different from “Simulation”. In lesson 4 you have seen Simulation, so when a model is simulated, it is executed on your computer. The other way in which a model like the ones in the previous pictures can be executed is by generating (from the model) an executable that runs (typically in real time) on the target platform (Arduino in our case), and unlike what happens when you simulate the model on the host computer, this executable will be downloaded on Arduino and runs on it. So you can’t connect for example a scope in a model that should be downloaded to Arduino to watch data, instead you should tell to Arduino to send datas to Serial Port (0) and you can watch it trought Serial Receive.

3.5 PID Example

As we said before, you can generate trought Simulink Arduino firmware without writing C code, and this extremely speeds up the prototyping phase.
An example of thi powerful tool is the following PID controller, created by using both Arduino Support Package, Simulink standard blocks, and ours Skyward Target for BLIMP.

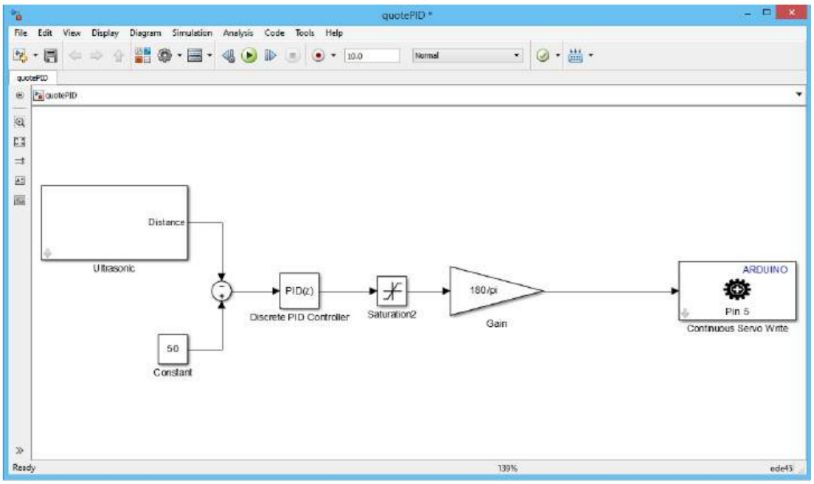

Figure 3.6: PID Example