EE302 Feedback Systems

Term Project Graphical User Interface and Arduino Code Description

Spring 2018

1. Scope

In order to control the angular position of the "servo arm", you will implement a controller that will run on the Arduino microcontroller. This controller will take the angular position error signal (desired angular position minus the measured angular position) and produce an actuating signal. The desired angular position is an internal variable in the microcontroller and the measured position is a discrete-time digital signal acquired by one of the analog-to-digital input ports of the Arduino board. The actuating signal will be converted to a PWM signal, which will be sent to the motor driver through one of the digital-to-analog output ports. If a PID controller is used, then the values of the P, I and D parameters are also kept as internal variables inside the microcontroller.

In order to identify the system parameters and then design the controller, certain signals must be observed. For this purpose, it will be helpful to observe the arm angle, error and actuating signals on a graphical user interface (GUI) running on a PC, both in real-time and offline. Another helpful feature of this GUI would be to easily set the internal variables such as the set point (desired angular position), and controller parameters. The communication between the PC and the Arduino can be satisfied by a serial communication through a USB cable.

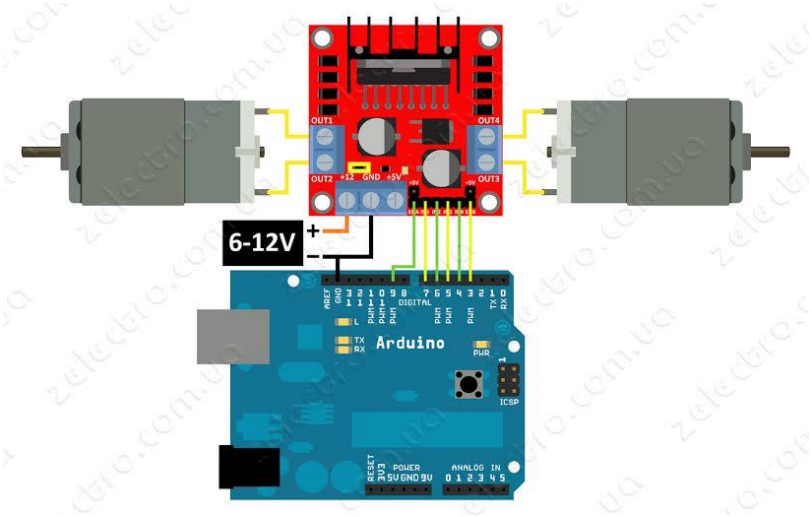
We supply you some code to help you get started for the objectives that are stated above. In the below, we describe the related software setup and give some guidelines.

2. Introduction

This document explains how to use the GUI designed for the EE302 Feedback Systems course. Please read the document from beginning to end before using or changing any codes. Do not worry you will only change a couple of lines in the Arduino code only. The GUI was designed to ease the tuning of the parameters for the controller and to visualize the signals such as error, arm angle, etc. The system consists of two parts. Python part and Arduino part. To use the Python code, you must have Python installed on your computer either on Windows or

Linux. Also, you have to install pip which will let us install some packages that we use in order to communicate and draw the GUI. You can use Python and pip via CMD and Git Bash on Windows. You have to use the terminal, if you are using any Linux distros or MAC. We will refer to this interface, CMD, Git BASH, terminal, as terminal throughout the document.

3. Hardware connections



First, you have to make your connections correctly. That is on Arduino pin 6 and pin 7 goes to the motor driver inputs and pin 9 goes to the enable pin of the motor driver. Arduino ground goes to a leg of the POT and Arduino 5V goes to the other leg of the POT. Arduino analog A0 pin goes to the middle leg of the POT. **Both Arduino and the motor driver have to share a common ground!**

4. Installation

Suggested software installation steps tested for a Windows 7 operating system.

1. Download and install Arduino 1.8.5 IDE (latest version with the default settings) from https://www.arduino.cc/en/Main/Software.
2. In order to make sure that your Arduino hardware and IDE is configured properly, you can run a simply blinking code. In order to do that, in Arduino IDE, go to “Tools” -> “Port” and set the correct port number. Open the example code in “File” -> ”Examples” -> ”Basics” -> ”Blink”. Then, click on the Upload icon. If everything is set up correctly, then the LED on Arduino should blink.
3. Download and install Python 3.6.3 (We have tried the system with this version) from https://www.python.org/downloads. (We have tested for

the 64-bit version). **Do not forget to check the option which will add PYTHON to your path while installation.**

4. Open command prompt. Type python. If it works, then no problem you can skip 5.
5. If it does not work, do the following to add Python to your path. Hold WIN key and press PAUSE key on your keyboard. Click Advanced System Settings. Click Environment Variables. Append your Path variable with the Python installation folder. Restart command prompt.
6. Open command prompt or use the one if there is already one open. Type pip. If it works then no problem you can skip 7.
7. Make sure that pip3.exe is also on your path which can be found in the "Scripts" subfolder under Python installation directory. If not, follow a similar step as in 4.
8. After installing both Python and pip, you can install the requirements that were shared with you through ODTUCLASS. Run pip3 install -r requirements.txt in your command prompt (or terminal).
9. A package called PyQt4 has to be installed. The installation on Ubuntu is straight forward, sudo apt-get install python-qt4 or you can check online, however, on Windows you have to download a .whl file from this link "https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyqt4". Download the version which suits with your python version ..--cp36.. for python 3.6.x.
10. After downloading the package do(You have to change the .whl file according to your system); C:\path\where\wheel\is\> pip install PyQt4-4.11.4-cp35-none-win_amd64.whl

5. Software setup and graphical user interface

The code that you will download from METUCLASS involves an Arduino code with .ino extension and two Python code files. The Arduino code will be (compiled and then) uploaded to Arduino board using Arduino IDE. Python code will run on your PC. The job of the Arduino code is to handle the serial communications and apply the PID controller parameters. The job of the Python code on the PC side is to visualize the data and input controller parameters. You can get more information about the details of the code by looking at the comments.

First, you have to upload your Arduino code to your Arduino board. Then, you can start the Python code which initializes the GUI. Type "python ee302_gui.py

–help” on the terminal and read the text. Here you have to declare the USB port that is used by Arduino and the window length. The window length describes the measurements to be drawn by the GUI at a time. It does not mean seconds. The default is 100. You can keep it as it is. The port however, must be declared as “—port <port name that is chosen from Arduino IDE>”. So that the command that you will enter at the end is similar to “python ee302_gui.py –-port COM4”. COM4 could be a different port according to your connection and can be found from Arduino IDE->Tools->Port. Python and Arduino codes are designed to communicate through the serial port. **Hence, it’s not possible to use the serial monitor of the Arduino IDE and the GUI at the same time. And you have to close the GUI to upload a new code to your Arduino board then, reopen the GUI.** After initializing the GUI, you will see a screen similar to Fig 1. Here Arm Angle and Error is straightforward to understand and the Motor Command is the pwm value that you give to your motor. The desired value is the desired angle. Direct is the direct pwm that you give to your motor without the controller. sumMAX is used in integral controller.

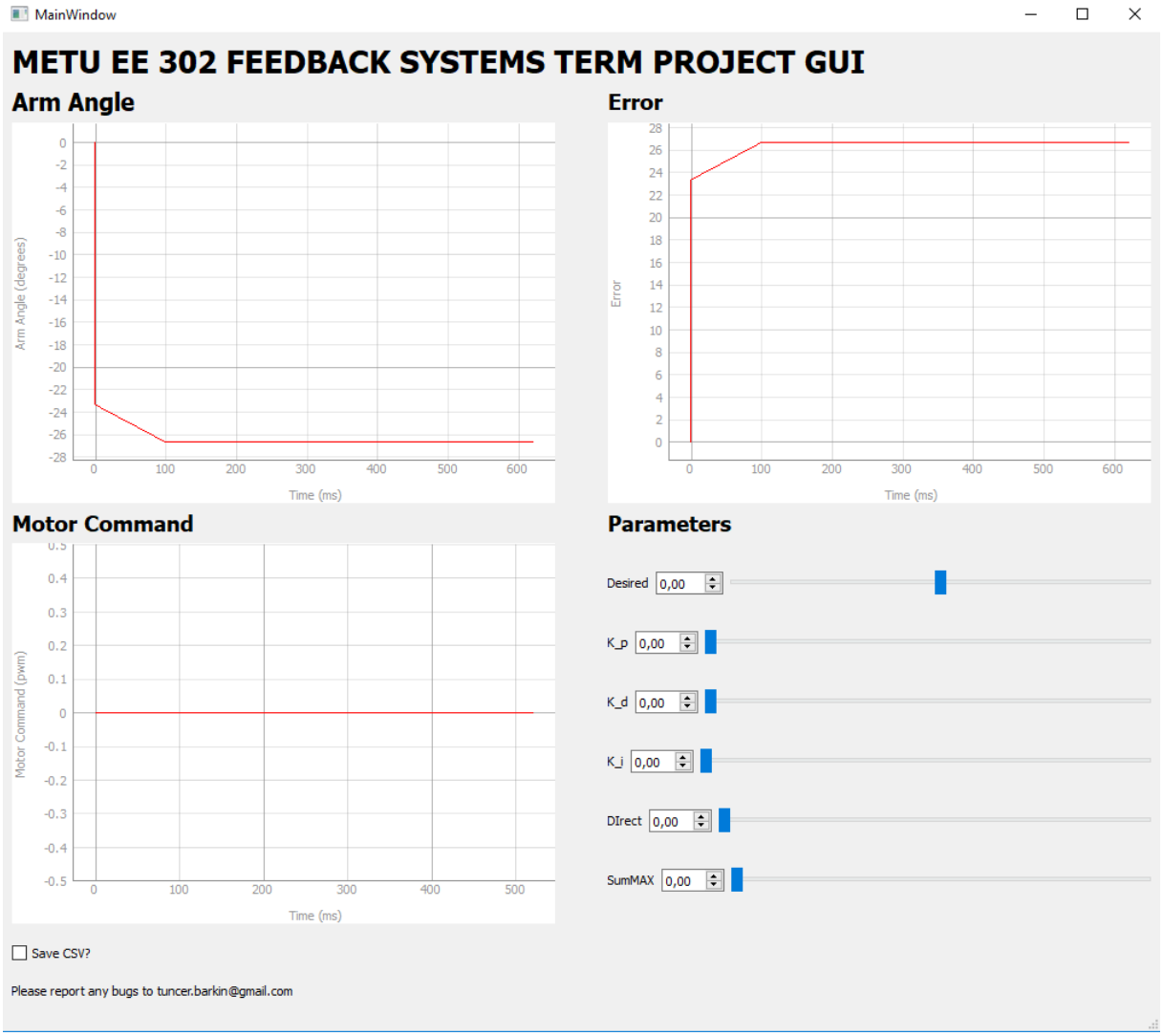


Fig 1 The GUI window

It is clear that you can change the controller values from the slider and also from the spin box that is located at the left of the sliders. Here you will see a “Save CSV?” box at the very left bottom. When you check this box, the GUI starts to save the data that it plots. Whenever you uncheck the box, the csv file will be saved to the folder “csv”. You can import your csv files by just dragging it to MATLAB workspace. The import interface will welcome you and you have to change the delimiter to comma. After that you can select the columns you want to import and click to Import Selection. The window should look like in Fig 2.

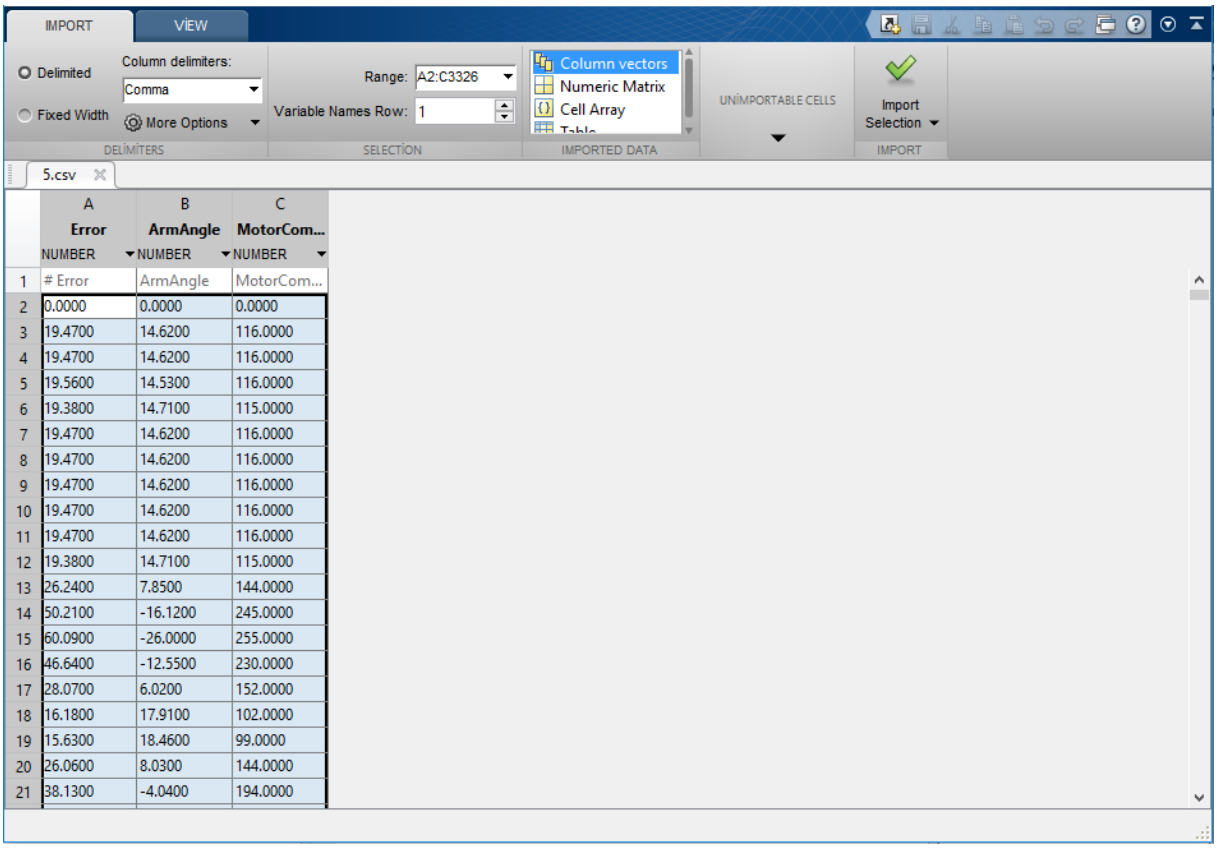


Fig 2. MATLAB import interface

6. Calibration of the arm angle measurement

Change the position of the arm and try to observe a change in the ArmAngle plot at the GUI. If it does not change, then it means there is something wrong which you have to take a look at. If you observe a change, you will see that the angle of the arm is not the true value. To solve this problem, bring your arm horizontal to the ground and keep it there. Uncomment the “Serial.println(ang);” line in your Arduino code and comment the rest of the Serial.print codes until the delay code. Now, you can observe the angle through the serial monitor of Arduino IDE without using the GUI. Adjust the two variables “angle_multip” and “angle_base” to achieve the zero angle measurement while holding the arm horizontal to the ground. You have to upload the Arduino code multiple times to achieve this. Also, try to tune these variables to have “-90” value of ang when you move the arm angle to -90 degrees. The “+90” case should be accomplished when you tune the variables for these two cases. However, you can still check if it works. You can also achieve this behavior from changing the line which is;

```
float ang = float (analogRead(angleSensorPin)*angle_multip)-angle_base;
```

You can also uncomment the line which is just above this code line and play with it to achieve your goal. That code is a different approach from this simple one. The commented line sum 3 measurements and then takes the average of these measurements to eliminate the noise. To conclude, you have to tune the code to have the angle value “ang” correctly.

After setting the arm angle correctly, do not forget to comment the “Serial.println(ang);” again and uncomment the lines that you have commented before. The final version of the serial prints should look like this in order to use the GUI;

```
//**Serial.println(ang);
Serial.print(t / 1000.0); // Sends the data to serial port.
Serial.print(" , ");
Serial.print(error); // Sends the data to serial port.
Serial.print(" , ");
Serial.print(ang);// Sends the data to serial port.
Serial.print(" , ");
Serial.println(motorCmd); // Sends the data to serial port.
delay(100);
```

As a final note, you can customize the PID controller such as the derivative term in the Arduino code, if you want to write your own code as long as you keep the variable names the same. The Arduino code is already commented according to that. So, the customizable codes can be seen easily.

Now you can begin working on your project. Have Fun!

If you have any questions, please send an email.

EE302 Assistants