5 degrees of freedom robot arm programming

Barbara Klojbert, Dominik Tál, Patrik Dósa

2018

University of Pannonia

Faculity of Information Technology

Department of Electrical Engineering and Information Systems

Engineering Information Technology Msc

Project Labor

5 degrees of freedom robot arm programming

Barbara Klojbert, Dominik Tál, Patrik Dósa

Supervisor: Dr. Attila Magyar

2018

Contents

List of Figures

1.
Introduction

Nowadays robots are not these futuristic and weird shaped machines as they were in the early 2000s American films. They are part of our life, and they make our days easier, more productive, and safer. It is pretty hard to define, what makes a robot, but we have some points which are true for all of them. First of all, we want to clarify, that our documentation will not say anything about software robots, we leave this topic to the IT security experts. The most important thing about machine robots is that they are programmable, and according to this program, it can make manipulations automatically. They can have external controller inputs, or they can have an embedded control. [2]

1.1 History of robots

Robots have their roots from the ancient ages, these cultures wanted to make automated devices for entertainment purposes, but mankind only had the proper materials and methods from the industrial ages, and in these ages they wanted to use robots in the industry to make the production cheaper and faster. As we reached the modern ages, engineers and inventors introduced automatic, remote controlled, and wirelessly remote controlled robots too. [1]

The word robot, comes from the Czech language, and it means “forced labor”. In its modern form it has been firstly used by Karel Capek’s play in 1920. The science of robotics is the engineering field which deals with the design, construction and operation of robots. The word of robotics has been introduced in Isaac Asimov’s Sci-Fi “Runaround” in 1942. In his books Asimov has written down the three laws of robotics. These are the following: [1]

1. ”A robot may not injure a human being, or, through inaction, allow a human being to come to harm.”

2. ”A robot must obey the orders given it by human beings except where such

orders would conflict with the First Law.”

3. ”A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.”

1.2 Modern robots

In every modern factory you can see robots. Robotic arms which assemble whole cars, robot trains which transport products to the warehouse, industry 4.0 is not the future, we live in it.

Why are robots good for mankind? They can make monotonous processes all day long without a lunch or a cigarette break. They can be used in hazardous areas without any health risks. To cut a long story short, they make everything more productive, safer, quicker. Their designers should be aware that they have to help humans work, not to take all of their work to make human workers useless.

According to industry 4.0 the separate robots and other machines are in a huge common network, and they synchronise their works, utilize the resources to reach the optimum. [2]

1.2.1 Types of modern robots

- Mobile robots
- Industrial (manipulator) robots
- Service robots
- Educational robots
- Modular robots
- Collaborative robots

1.2.1.1 Industrial (manipulator) robots

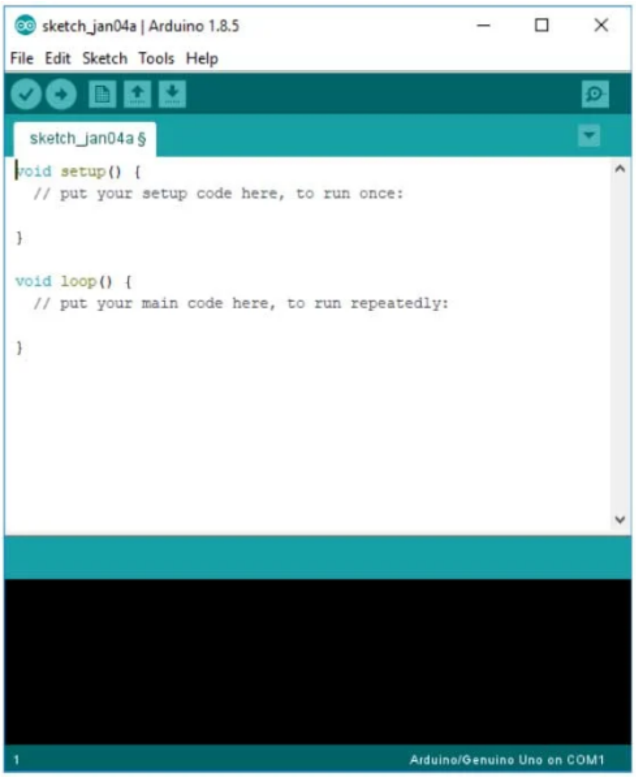
Industrial robots are used for manufacturing, they are programmable and automated. They can move on two or more axes, but they are not able to change their physical position.

Their fundamental is the same, this structure has to be able to move a specific tool to the certain position, and hold it there until the manipulator finished its work. This manipulator can do welding, painting, screwing, lifting, etc. [2]

2.
Applied devices

2.1 Arduino IDE

We used Version 1.0.6 of the Arduino IDE to program the Arbotix-M microcontroller. The Arduino IDE is an open-source software which can be used to implement, compile and upload our codes to the microcontroller. The IDE can run on Linux and Mac OS as well as on Windows. Everyone can use the IDE to program any Arduino or Arduino based microcontroller like the Arbotix-M. The environment was created in Java and with other open-source and free software. Arduino has lots of advantages like: easy way to use, huge community who use Arduino, so they can help through online communication. [5] [6]


sketch_jan04a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_jan04a §
1
Arduino/Genuino Uno on COM1

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 2.1 Arduino IDE [7]

2.2 Arbotix-m Robocontroller

This is a versatile Arduino based robot controller method for systems which based in BIOLOID and DYNAMIXEL. Moreover this controller allows direct connection between the Dynamixel network and the board. ArbotiX-M Robocontroller has lots of advantages, as another Arduino compatible microcontroller, for example open source community of libraries and examples. The main difference between Arbotix and Arbotix M are the following:

- In name of Arbotix M m stand for mini, cause it's smaller than Arbotix.
- A smaller footprint(61mm X 61mm)
- 10 more general purpose I/O pins
- Barrel Jack for DC Power
- Separate Power Bus for 4 PWM Digital outputs for Hobby Servos
- No dedicated I2C header (I2C is still available on D16 = SCL D17=SDA)
- No DC Brushed Motor Controller
- Not compatible with the RX-Bridge

So the Arbotix-M has the following properties:

- 16MHz processor (ATMEG644p)
- 2 serial ports, these make easy to program using FTDI to USB connection.
- 3 TTL Dynamixel network interfacing ports (for direct motor connection)
- Analog and digital pins

Last but not least arbotixM robocontroller has a big disadvantage. Usually we said: Arbotixm is cheaper compare to the overall price of robots. Because of this, it's not got the fastest processors or the best memory.

2.3 Pincher Robotic Arm

The robot arm is a 5 degree-of-freedom robotic arm. This means the robot arm can can rotate along 5 different axes. [5]

Pincher Robotic Arm Specification	
Weight	550G
Vertical Reach	35CM
Horizontal Reach	31CM

2.3.1 Servo motor

We have used Dynamixel AX-12A Robot Actuator servos in our project. The are powerful, durable and they have diagnostic functions too. They can monitor their voltage, and it is very important in protecting our battery from the damages of a critical undervoltage situation.

These servos can measure their rotating angle, so they can be controlled very precisely. [3]

Servo motor specification	
Weight	53.5g (AX-12/AX-12+), 54.6g (AX-12A)
Dimension	32mm * 50mm * 40mm
Resolution	0.29°
Gear Reduction Ratio	254 : 1
Stall Torque	1.5N.m (at 12.0V, 1.5A)
No load speed	59rpm (at 12V)
Running Degree	0
Running Temperature	-5°C ∼+70°C
Voltage	9 ∼12V (Recommended Voltage 11.1V)
Command Signal	Digital Packet
Protocol Type	Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
Link (Physical)	TTL Level Multi Drop (daisy chain type Connector)
ID	254 (0-253)
Communication Speed	7343bps ∼1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Material	Engineering Plastic

2.3.2 Pincher tool

Our robot arm uses a so called pincher tool. Its maximum width is <> centimeters. It can produce a <> N holding force when closed. This tool is controlled by the fifth servo of the system, and it is made from industrial plastic too. [3] [4]

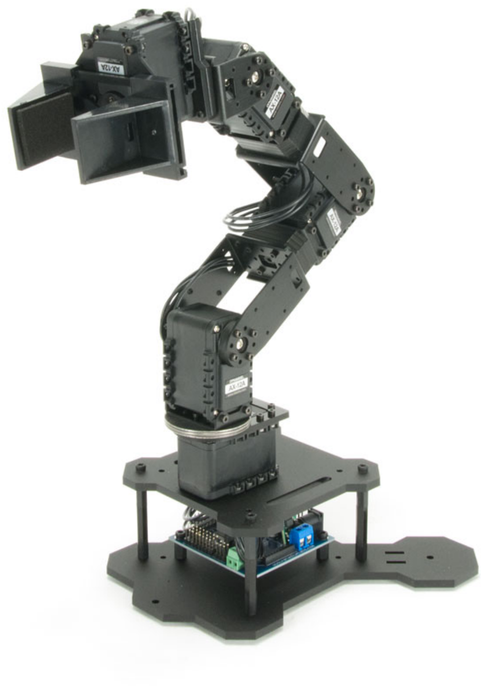


Figure 2.2   Pincher [4]

2.4 Programming language

Arduino doesn't run either C or C++. It runs machine code that compiled from C, C++ or any other language. The Arduino IDE accepts C and C++ as it is. Most of the libraries are written in C++, but it is very different from most C++ varieties. The Arduino C++ language has a lot of abstraction built in.

Most of the embedded systems in the microcontroller world are using C++ language. Arduino code's core functions are simply a set of the C++ classes and libraries. Arduino language simplifies a lot of things, but we can still write our functions as we want and where we need more specialization.

3.
System design

3.1 Model

3.2 Systems

4. Test

Our first plan was to test all of the servos and the robot's possibilities. We reached that because we implemented a test software which can communicate with the laptop on serial.

The program firstly set all the servos to center and after that relax all of them.The software's input is the servo number and the requested position (from 0 to 1023) and after that the robot's servo motor set that parameter to the selected servo. With this we could test all the servos. The basic code is the following:

```
#include <Commander.h>
#include <ax12.h>
#include <BioloidController.h>

void setup() {

  //open serial port
   Serial.begin(9600);
   delay (500);
   Serial.println("###########################");
   Serial.println("Serial Communication Established.");

  // set everything to center
    Serial.println("###########################");
    Serial.println("Set all servo motor to center (512).");
    for (int i = 1; i <= 5; i++){
      SetPosition(i, 512);
      delay(100);
    }

    Serial.println("###########################");
    Serial.println("Relax all servo motor.");
    for (int i = 1; i <= 5; i++){
      Relax(i);
    }

}

void loop() {

  // read the servo motor number
    Serial.println("############################");
    Serial.println("Which servo motor?");

    while(!Serial.available()){} // wait til the serial input
    char servoInByte = Serial.read();
    int servoNumber = servoInByte - '0';

    unsigned int positionValue=0; // Max value is 65535
    char incomingByte;

  // read position value
    Serial.println("############################");
    Serial.println("Which position?");


    while(!Serial.available()){}
    delay(100);
    if (Serial.available() > 0) { // something came across serial
      positionValue = 0;       // throw away previous integerValue
      while(Serial.available() > 0) {
        incomingByte = Serial.read();

        // Serial.println(incomingByte); // just for test

        positionValue *= 10; // shift left 1 decimal place

        // convert ASCII to integer, add, and shift left 1 decimal
            place
        positionValue = ( (incomingByte - '0') + positionValue);
      }

      SetPosition(servoNumber,positionValue);
      Relax(servoNumber);

      Serial.print("Servo");
      Serial.print(servoNumber);
      Serial.print(" in Position: ");
      Serial.println(positionValue);
    }
}
```

5.
Summary

The first milestone of the project has been reached successfully, our main goal was getting to know this very complicated system and design the plans of its advanced control software. The biggest challenge is implementing the mathematical model of a three-dimension coordinate system on this manipulator. Fortunately, Arduino IDE is one of the simplest and easily usable development tools.

We think that our method is very effective with a very early constructed test for the software. This is not a pure example of TDD (Test Driven Development), but it has its own advantages too, and it is efficient enough for this project.

We started to discover a very interesting and very popular field of robotics. Robot manipulators are the most common robots all over the world. They are used in ever modern factory. They have the oldest origins, but with Industry 4.0 they have developed rapidly, and they have many more opportunities in the future. Robot manipulators ca be integrated into other types of robotic system, for example on the board of a mobile robot, or a manipulator can be part of a co-robot system.

5.1 Future plans

In the second part of the semester our goal will be to finish our planned work, implement the solution of the mathematical model, write the documentation, and specify system tests. Hopefully, after the execution of the tests we will report that everything is passed, and the robot manipulator will work safely.

We have further plans for the system and for other semesters, the movements of the robotic arm might be logged by other Arduino add-on devices, and this log files could be used for developing the software of the arm.

Bibliography

[1] Robotics,
`https://www.techrepublic.com/article/`
`robots-of-death-robots-of-love-the-reality-of-android-soldiers-and-why-la`

[2] Tom Harris
Ḧow Robots WorkḦow Stuff Works.

[3] Ax actuator,
`http://support.robotis.com/en/techsupport_eng.htm#product/`
`dynamixel/ax_series/dxl_ax_actuator.htm`

[4] Pincher Robot Arm,
`https://www.trossenrobotics.com/p/PhantomX-Pincher-Robot-Arm.`
`aspx`

[5] Daniel Jacobs,
Introduction to dynamixel Motor Control Using the ArbotiX-M Robocontroller

[6] Arduino IDE
`https://en.wikipedia.org/wiki/Arduino_IDE`

[7] Arduino IDE,
`https://www.digikey.com/en/maker/blogs/2018/`
`introduction-to-the-arduino-ide`

[8] Pincher,
`https://www.trossenrobotics.com/Shared/Images/Product/`
`PhantomX-Pincher-Robot-Arm-Kit-Mark-II/pincher.jpg`