# Robot Arm

Website: <https://github.com/voltAG3D/CENG-317--Daniel>

Names: Daniel Shelepinsky, Ali Khaliq and Matthew Gelfand

Course: Computer Engineering Technology

Date: April 23rd 2018

# Declaration of Joint Authorship

The robotic arm project consist of three group members (Daniel Shelepinsky, Ali Khaliq and Matthew Gelfand) and is a joint effort on the completion of the project. The work for the project has been divided equally among the group members. Ali Khaliq and Matthew Gelfand has worked on the hardware aspect of the project; which includes connecting an Arduino with a Raspberry Pi, controlling the servo motors. Daniel Shelepinsky has worked on the design and functionality of the mobile application. Daniel Shelepinsky has also worked on the database in terms of setting it up, connecting it with the app and maintaining it. Testing and maintenance for the hardware and software has been tested by all three members equally and all project updates and changes have been checked and approved by all members.

# Approved Proposal

September 2017 and reapproved January 2018

Prepared by Daniel Shelepinsky, Ali Khaliq and Matthew Galfand

## Executive Summary

As a student in the Computer Engineering Technology program, I will be integrating the knowledge and skills I have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the software app that will connect to a hardware as well as to a mobile device application. In the app, we will have incorporated a controller that will be used to control the hardware. The database will store the coordinates of the robot arm and the angles that the motor and arms need to be when it returns to rest position. The mobile device functionality will include some very basic test functions and commands to move the arm. It will create logs and save that data, so the developer can later look it at and make improvements to the hardware or software and other users can use these logs to reproduce experiments. In the winter semester I plan to form a group with the following students (Daniel Shelepinsky, Ali Khaliq and Matthew Galfand), who are in a different course. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 3 student group.

# Abstract

The purpose of the Robot Arm project is to create a robot arm that has many purposes but for the main purpose would be an assembly line or sorting. The main components are servo motors, color sensor, mobile app, and the robotic arm will all be explained in great detail. The arm itself will be controlled by four analog feedback servo motors. One at the base of the arm to control left and right in a 180 ° Than two more for the arm itself to extend it and retract. The last motor controls the grip which will open and close it. Than all servo motors will report back to the database their positions. Which the user can edit through the mobile application. The arm itself is made out of acrylic parts. It will be connected to a Raspberry Pi 3 and an Arduino. The Pi being handling all software aspects of the project and the Arduino handling the PWM. The main focus of this technical report is to give the users a full understanding of the project and how it came to be.

# Table of contents

Declaration of Joint Authorship

Proposal

Abstract

Illustrations or Diagrams

1. Introduction
2. Project Description
   1. Requirements specifications
      * 2.1.1 Mobile Application Concept
      * 2.1.2 Application Screen
      * 2.1.3 Hardware Interfaces
      * 2.1.4 Software Interfaces
      * 2.1.5 Communication Interfaces
   2. Build Instructions
      * 2.2.1 Introduction
      * 2.2.2 Bill of Materials/Budget
      * 2.2.3 Time Commitment
      * 2.2.4 Mechanical Assembly
   3. Specific Requirements
      * 2.3.1 Database
      * 2.3.2 Application
      * 2.3.3 Hardware
   4. Progress Reports
   5. Problem Encountered
3. Conclusion
4. Appendix
5. Reference

# Introduction

Our hands can only do the same task over and over without stopping to take a break. A robot arm when programmed and built correctly can fix this issue and allow the user to repeat the same task over and over and does not need to stop for breaks like a human arm does. If any injuries were to happen on the job. It would happen to the robot arm and not to the human/user which would mean reduced work place injuries where the arm can be applied to. The arm itself is cheap and easy to setup and can easily replace the parts if the arm does break down. This would mean production in a company would not fall as much as if a human would get injured. The arm can be automated or controlled from a distance with a mobile application which would again leave the user free of harm. The arm does not require rest so it can operate all day and night if need be.

The Arm consists of four motors and a color sensor. The four motors will control the arms and movement going from left to right and extension/retraction. The color sensor is used to detect objects by color and can be programmed to ignore certain objects or pick them up and sort them depending on what the user wishes the arm to do. If the user does not like to use a mobile application or automation they can use buttons/controller that will move the arm if they so prefer. But the controller will have to be set up by the user.

# Project Description 2

## Requirements specifications 2.1

### Mobile application Concept 2.1.1

The Concept for the mobile application is to control the Robot arm from afar with either a phone, iPad, computer all depending on the user’s choice. The application will allow the user to move and report positions of the arm and possibly debugging if the arm ever gets in trouble/stuck. The application also includes a database which stores information about the arm and positions that the arm is currently in. The user can also move the arm by changing the values in the database but that is roundabout method and not advised.

### Applications Screen 2.1.2

There is only one screen which is the control screen and all the communication is done in there. The control screen has 4 seek bars which all have limits set so they do not hit each other and cause damage to the arm as the motors do not move slowly the move at full speed to get from point A to B. Each seek bar controls a motor that is labeled, Claw, Elbow, Shoulder and base and will move these according to the limits that have been set. The user can set the limits if they wish too but they will have to edit the android studio application directly. At the top right of the screen it shows the positions that the arm is currently in and changes upon the arm moving so it will always show the current the position of the arm.

## Hardware Interfaces 2.1.3

The only hardware interface is the Raspberry pi or Arduino depending on which on you wish to use. For this project we used the Raspberry pi so the interface of the pi is like a normal computer so it is relative easy to navigate and use. There are many guides online to learn how to use them if need be. The pi will have a python code which will grab from the database and send it to the arm.

## Software Interfaces 2.1.4

The software interface is just the mobile application which sends signals to the database than the pi pulls and sends it to the robot arm. The pi needs to be running the python code at all times in order for the connection to work.

# Build Instructions 2.2

## Introduction

The robot arm can be used for many things as long as it is programmed and built accordingly. Our arm will be used for basic use and testing and just learning how to use robotics. Meaning that it will be able to move with 4 motors pick up and drop items and sort items through color. The Arm will include a phone application to control the arm and store information into a database if the user so wishes to have.

## Budget

First you will need a raspberry pi or something to be able to control the motors.

* <https://www.amazon.ca/gp/product/B01CCF6V3A/ref=oh_aui_detailpage_o02_s00?ie=UTF8&psc=1> This includes everything that will be needed for the arm to work GPIO pins, WiFi, Bluetooth etc. This goes for $99 CAD

Next on the list is the motors analog feedback micro servos bought from adafruit or you can use full size motors.

* <https://www.adafruit.com/product/1449> These go for $10 CAD a piece which is $40 CAD. 4 wires that come with these motors. 1 white for the feedback and the usually power, ground and signal.
* [https://www.adafruit.com/product/1404 These go for $15](https://www.adafruit.com/product/1404%20These%20go%20for%20$15) CAD a piece which is expensive but they are larger and more stable than the micro servos. Also have the same wiring as the micro.

Here you have a choice of either getting the arm chassis laser cut or buy it. I got it laser cut via the Humber prototype lab but in case you wish to buy it you can from

* <https://shop.mime.co.uk/> Which is around 50-70 $CAD

Total budget: 235 $CAD is how much we spent on this project because we had to rebuy some items and we bought some useless items that turned out we didn’t need in the end.

## Time Commitment

The time commitment will change depending on how skilled or new you are to building, setting up and controlling motors with code. So with that in mind please set a time you think is good for you. The times below is how long it took us.

* Buying the materials and awaiting for them to arrive will take about 1 week – 2 weeks depending on where you order I would suggest amazon and ad fruit as they are pretty quick with their orders.
* Assembly is pretty easy when steps are followed: <http://www.instructables.com/id/Pocket-Sized-Robot-Arm-meArm-V04/>. The parts are easy to break but are easy to replace as well as long as you have the AutoCad files. 1 hour – 3 hours
* Wiring can be hard as the raspberry pi has a floating voltage which can do damage to your motors if you are not careful or if you have another power source. Before hooking up the motors to the arm be sure to test the motors and see if they work. 1 hour - 2 hours
* Coding is the hardest part out of the project as it is the backbone. It connects the arm to the pi and the database as well as the mobile application which overall will move the arm.

Total time: 1 Month to setup, build

1 week to debug and get it to functional work

1 week to have it fully functional.

## Mechanical Assembly

Once you have all your parts you may proceed. Side note, always double check your wiring and set the motor limits before you put them into the arm. Also select which motors you wish to use either the regular or micro as this will affect the overall design. Moving forward you will have to have the arm parts and everything near by to start.

Step One: confirm which GPIO pins you wish to use. I used 12,13,18,19 for PWM and 2 GPIO pins for each motor: 16, 20; 21, 5; 6, 26 and 23, 24.

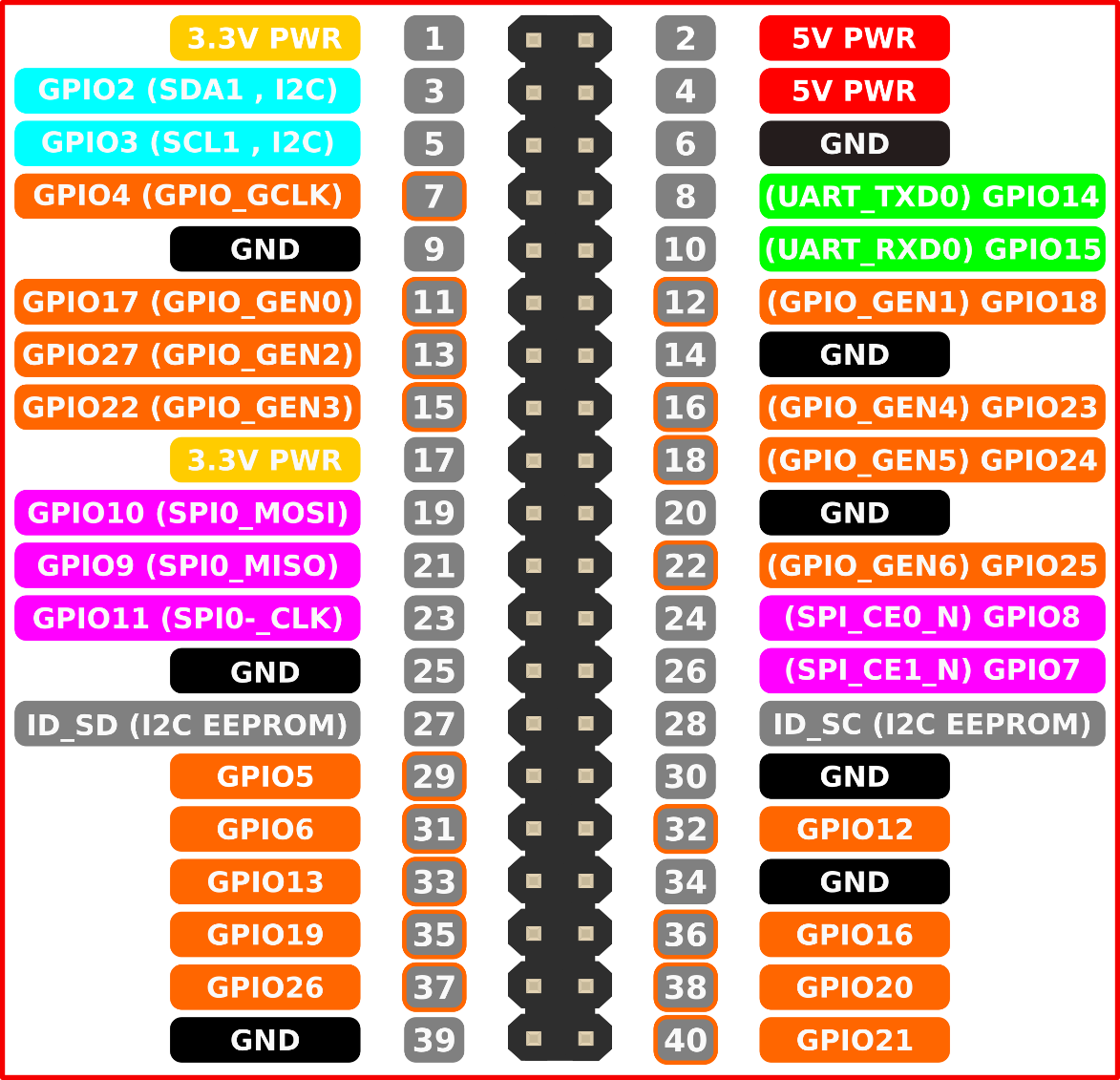
Step Two: Once you have your pins selected connect them to your motors. Then test your motors by running a simple script with either C or Python. The Code I used for the arm is here <https://pastebin.com/nCWX3j4F>. I also suggest adding LEDS to show that they are getting power in case. Before going to step 3 really make sure every motor you have works.

Step Three: Build the Arm using the guide given by <http://www.instructables.com/id/Pocket-Sized-Robot-Arm-meArm-V04/>.

Step Four: The arm is ready to be used but always test it and make sure it works intended and double check and everything should be ready to go.

## basicImage dump

Basic way to connect a motor to a raspberry pi.

GPIO pins used by a raspberry pi 3. Some will have to be set through code.

# Project Breakdown 2.3

## Database 2.3.1

The Database will be firebase. The reason behind this is because it is free and a very friendly database for newcomers to work with. It only has 1 table which includes and will store the following information: Claw motor, elbow motor, arm motor and base motor. This also controls the overall communication of the robot as it pulls the data from the mobile application and then pushes that data to the raspberry pi which than from the pi pushes to the arm.

## Application 2.3.2

The overall concept of the mobile application for the project is to control the four servo motors of the robotic arm. There will be no log in as there is no point to have a user logs option. So there are only two major parts that will be included within the application and that will be the actual control of the motors and the connection to the raspberry which will overall control the arm.

The first option of the application is to control the motor with four sliders that will control the arms movements. When the application first begins, the sliders will be set to a default (centered) position. When exiting the application, the robotic arm will return to its default position or when the application boots up again it will also reset to its default position. There is a reset option within the application to set it to default as well.

The Second option is the connection. There are currently three options to connect with. One being IP and port. The port being provided by the pi but before you can do this you will need to set up the pi as a server. The second option is Bluetooth the pi3 comes with Bluetooth built in so you will not need any extra parts. The third being database which require the user to do nothing but open the application and run the python code on the PI as the connection is through the database. The other two connections are not visible as they will need to go into the code in android studio and edit it to be visible and usable by the user.

## Hardware 2.3.3

Robotic Arm: The arm was created by ME industries. I have taken that design and added a database along with a phone application and switched out the motors for analog feedback servos.

Analog feedback servos: These servos are just like any other but have another wire that is wire which reports back on the position it is in with analog.

Optional micro analog feedback servos: These are just small analog feedback servos if the user wishes to save and use smaller motors.

Raspberry pi3: This is the main component as it will be sending the signals to the arm in order to move it.

Wires, Resistors, Buttons: You will need wires in order to connect the arm together and have it work from the pi. As it cannot send signals without these. The buttons are optional if the user wishes to control from a controller but the user will have to set up it up themselves.

# Progress Reports 2.4:

The progress reports show the 3 main reports that we had to do over the semester and each with a goal in mind. The first progress report is just a generally idea of where we currently are in the project. The second progress report is the integration between the group member’s projects and how far it is gotten. The final progress report is trouble shooting and errors that we are currently running into and how far we have gotten.

The point behind these reports is to give yourself deadlines to reach and attempt to reach on time so that you don’t fall behind your schedule.

### Progress Report 1:

Hello Sir,

Over the pass weeks not much has actually happened in regards to work besides writing up the report and coming up with ideas for what to do.

- We have gotten a new arm. (laser cut instead of 3d printed)

- finished the starting points of the report.

- Come up with ideas to make it unique.

- Working on Hardware build instructions

Problem/Question: If possible can i still switch projects/Sensors? The sensor to which we would switch to is a TCS3200 color sensor. Our reasons to switching to this is because there have been many robot arms done in the past and we would like something more unique.

Financial: We have bought a new Sensor which is the TCS3200, Screws and bolts and an Arduino.

Regards,

Daniel Shelepinsky

N01019434

### Progress Report 2:

Dear Kristian

The integration between our groups is going along fine so far as the application, database and pi can talk to each other to an extent. The next step(s) is to get the application to control the motors themselves, redo the wiring of the arm, possibly make a PCB and or get a box-base to hold it all under so it isn’t in view. There has been a minor setback and that is that we tightened the motor to the claw too much and it broke as a result but that has already been replaced we just need to attach the new claw.

Regards,

Daniel Shelepinsky

### Progress Report 3:

Hello Sir,

We have run into some errors so far mostly with the hardware but that is getting resolved by Ali and Matthew. As for the ones in the software. We have run into errors regarding the connections and the like but we have solved this and saved us a great amount of time by using the database to connect between the arm and the application. Which we have semi working but it does not send signals to the arm as of yet. Another thing we had to ditch was PHP. The reason for this was we cannot send PWM signals with PHP and we could do connect to the database via python and send signals with python.

Lastly the buttons will be removed. All that is to get the pi to send signals from the application to the arm and have it move.

Regards,

Daniel

# Problems encountered 2.5

During the creation of the arm we had to go through a great many tests to ensure things have worked. Such as motors now these gave us a few problems because we thought that micro servos were the same as normal servos and fed it the same voltage as regular servos which burnt them out. Which set us behind as we had no backup’s motors to fill the slots. After testing the new motors, we have learned their limits and how to use them with code. Of course, we ran into issues with the coding as we are new to python but after reading up on a few guides online on how to use the GPIO signals to send PWM to the motors. After the coding was done we than moved onto mobile application which would control the arm now at first, we had issues selecting how to do this as there are many options of connecting a phone application to a raspberry pi such as bluetooth, server and database. We ended up using the database as it seemed the easiest way of connecting from the phone to the pi. The last issue we ran into was the motors again after the arm could not support its own weight we had to think of a way to fix this so we upgraded the micro servos to regular analog feedback servos. Which than caused us to change the design of the robot arm all together and as the semester was coming to an end. We ended up rushing the design and had some minor flaws as the hole to fit the motor wasn’t an exact fit but it still worked.

## Conclusion 3

This project was selected because the overall idea of robotics is a cool idea to us and can be used for a multitude of applications. Such as working in places that would harm a human and allow them to work from a safe distance. Like working underwater or in extreme heats. As long as the arm has correct materials for the job it will work as intended. The amount of uses for a robotic arm are as much as a human arm. Eventually the robot arm or really any part of a human can be replaced by robotics parts. Now our arm is a simply arm to help us understand the basic use of motors and an overall learning experience and introduction into basic robotics.

# Appendices 4

Python test code

import RPi.GPIO as GPIO

import time

positionOne = 7

positionTwo = 7

positionThree = 7

positionFour = 7

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.OUT)

GPIO.setup(19, GPIO.OUT)

GPIO.setup(12, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

GPIO.setup(16, GPIO.IN)

GPIO.setup(20, GPIO.IN)

GPIO.setup(21, GPIO.IN)

GPIO.setup(5, GPIO.IN)

GPIO.setup(6, GPIO.IN)

GPIO.setup(26, GPIO.IN)

GPIO.setup(23, GPIO.IN)

GPIO.setup(24, GPIO.IN)

GPIO.setwarnings(False)

counter = 0

first = GPIO.PWM(19, 50)

second = GPIO.PWM(13, 50)

third = GPIO.PWM(18, 50)

fourth = GPIO.PWM(12, 50)

first.start(7)

second.start(7)

third.start(7)

fourth.start(7)

time.sleep(3)

first.start(0)

second.start(0)

third.start(0)

fourth.start(0)

stop = True

while stop:

oneMovU = GPIO.input(16)

oneMovD = GPIO.input(20)

twoMovU = GPIO.input(21)

twoMovD = GPIO.input(5)

threeMovU = GPIO.input(6)

threeMovD = GPIO.input(26)

fourMovU = GPIO.input(23)

fourMovD = GPIO.input(24)

GPIO.setwarnings(False)

if(oneMovU):

if(positionOne != 11):

positionOne = positionOne + 1

first.start(positionOne)

time.sleep(1)

first.start(0)

print("Moving level 1")

print("Waiting Button Release")

while oneMovU:

oneMovU = GPIO.input(16)

elif(oneMovD):

if(positionOne != 0):

positionOne = positionOne - 1

first.start(positionOne)

time.sleep(1)

first.start(0)

print("Moving Level 1")

print("Waiting Button Release")

while oneMovD:

oneMovD = GPIO.input(20)

elif(twoMovU):

if(positionTwo != 11):

positionTwo = positionTwo + 1

second.start(positionTwo)

time.sleep(1)

second.start(0)

print("Moving Level 2")

print("Waiting Button Release")

while twoMovU:

twoMovU = GPIO.input(21)

elif(twoMovD):

if(positionTwo != 0):

positionTwo = positionTwo - 1

second.start(positionTwo)

time.sleep(1)

second.start(0)

print("Moving Level 2")

print("Waiting Button Release")

while twoMovD:

twoMovD = GPIO.input(5)

elif(threeMovU):

if(positionThree != 11):

positionThree = positionThree + 1

third.start(positionThree)

time.sleep(1)

third.start(0)

print("Moving Level 3")

print("Waiting Button Release")

while threeMovU:

threeMovU = GPIO.input(6)

elif(threeMovD):

if(positionThree != 0):

positionThree = positionThree - 1

third.start(positionThree)

time.sleep(1)

third.start(0)

print("Moving Level 3")

print("Waiting Button Release")

while threeMovD:

threeMovD = GPIO.input(26)

elif(fourMovU):

if(positionFour != 11):

positionFour = positionFour + 1

fourth.start(positionFour)

time.sleep(1)

fourth.start(0)

print("Moving Base")

print("Waiting Button Release")

while fourMovU:

fourMovU = GPIO.input(23)

elif(fourMovD):

if(positionFour != 0):

positionFour = positionFour - 1

fourth.start(positionFour)

time.sleep(1)

fourth.start(0)

print("Moving Base")

print("Waiting Button Release")

while fourMovD:

fourMovD = GPIO.input(24)

else:

if counter > 100000:

counter = 0

print("waiting....")

counter = counter + 1

# References 5.

1. (2017, October 14). Pocket Sized Robot Arm MeArm V0.4. Retrieved March 26, 2018, from <http://www.instructables.com/id/Pocket-Sized-Robot-Arm-meArm-V04/>
2. Adafruit's Raspberry Pi Lesson 8. Using a Servo Motor. (n.d.). Retrieved April 22, 2018, from <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-8-using-a-servo-motor/overview>
3. Controlling DC Motors Using Python With a Raspberry Pi. (n.d.). Retrieved from <https://business.tutsplus.com/tutorials/controlling-dc-motors-using-python-with-a-raspberry-pi--cms-20051>
4. Vázquez, L. A., Jurado, F., Castañeda, C. E., & Santibáñez, V. (2016). Real-time decentralized neural control via backstepping for a robotic arm powered by industrial servomotors. IEEE Transactions on Neural Networks and Learning Systems, PP(99), 1–8. https://doi.org/10.1109/TNNLS.2016.2628038
5. Wu, J., Law, M. K., Mak, P. I., & Martins, R. P. (2016). A 2- µW 45-nV/ #x221A;Hz readout front end with multiple-chopping active-high-pass ripple reduction loop and pseudofeedback dC servo loop