International Journal of Engineering Research and General Science Volume 3, Issue 6, November-December, 2015
ISSN 2091-2730

Anthropomorphic Robotic Arm

Parasdeep Singh [i], Prashant Saxena [ii]

[i]Independent Reasearcher, parasdeep@protnmail.ch -9921674762

[ii]Independent Reasearcher, prashant.saxena@sitpune.edu.in-7387492617

Abstract- Our project basically deals in mechatronics, which aims to integrate mechanical, electrical and biological systems. In this report we will look at development of an anthropomorphic robotic arm that will mimic actions of a human arm.

Search and Rescue vehicle are used for a myriad of tasks in the 21st century. They are designed to do particular tasks that humans can but will not do, or even things humans cannot do. Our objective is to design and construct a vehicle to be used in an emergency response mission that will be able to venture into terrain and environments that are otherwise too dangerous for human responders.

The main function of the Rescue vehicle is to detect survivors using real time video transmission and send robot location via Bluetooth GPS receiver mounted on robot. The video camera will also be connected to PC wirelessly for real time video transmitting. The GPS will also send location data to PC and it will have the location where rescuers are to be sent. The vehicle is manually controlled by an operator and will be driven via looking at real-time video feedback received on PC.

The Mechanical Engineering part of this project is to design the actual platform for the robot to travel. Vehicle has 6 wheels, each wheel powered individually by an electric motor similar to many bomb disposal robots, so that it can traverse rough terrain. Motors with appropriate torque and speed have been selected based on weight and performance parameters. Two basic factors have been considered while designing the vehicle, first is vehicle performance and second is cost.

We aim to create an affordable robotic arm that is lightweight, easy to use, and performs as similar as possible to a natural arm. Our first task is to detect myoelectric signals from hand movements, process them and convert them to a form suitable as input to a microcontroller. We also plan to make a functional elbow joint and a palm with fingers capable of independent movement, minimizing weight and construction cost while maintaining structural strength.

Also, in order to improve its user efficiency, we have mounted the slave arm on top of a vehicular mobile unit. This enables us to move the slave arm to different locations too to perform multiple tasks and to even reach to areas which are difficult for a human physically. This vehicular mobile unit is remote controlled and is very user friendly. It has been designed keeping in mind all the needful and expected ergonomics and can even bear the weight upto 50kg for a possible situation where the slave arm is expected to lift something heavy and move it. This vehicle also has the ability to cross considerable size of obstacles without getting disbalanced or toppling.

Constraints like weight, dimensions, sensitivity of the arm and mobile vehicle will be verified at each stage of development.

Keywords- Mechatronics, microcontroller.Bionic arm,master slave, potentiometer, degree of freedom, servo motor ,Aurdino.

INTRODUCTION

Our project basically deals in mechatronics, which aims to integrate mechanical, electrical and biological systems. In this report we will look at development of an anthropomorphic robotic arm that will mimic actions of a human arm.

We aim to create an affordable robotic arm that is lightweight, easy to use, and performs as similar as possible to a natural arm. Our first task is to detect myoelectric signals from hand movements, process them and convert them to a form suitable as input to a microcontroller. We also plan to make a functional elbow joint and a palm with fingers capable of independent movement, minimizing weight and construction cost while maintaining structural strength.

Also, in order to improve its user efficiency, we have mounted the slave arm on top of a vehicular mobile unit. This enables us to move the slave arm to different locations too to perform multiple tasks and to even reach to areas which are difficult for a human physically. This vehicular mobile unit is remote controlled and is very user friendly. It has been designed keeping in mind all the needful and expected ergonomics and can even bear the weight upto 50kg for a possible situation where the slave arm is expected to lift something heavy and move it. This vehicle also has the ability to cross considerable size of obstacles without getting disbalanced or toppling.

Constraints like weight, dimensions, sensitivity of the arm and mobile vehicle will be verified at each stage of development.

CHAPTER 2: LITERATURE SURVEY

2.1 ROLE OF ELECTRONICS ENGINEERING

The role of electronic engineers in this project is to detect and receive the signals from the master arm .The signals sent by the master arm are carried through to the slave arm. The concept is that when movement is made in master arm, there is a voltage variation of some order in the connected potentiometers. This variation can be easily sent to the slave arm to copy the actions. The signals once detected have to be treated and transmitted using a micro controller.

We have used potentiometers which will be attached to the master arm. These will then send the signals and these signals are interpreted by the microcontroller which drives the servo motor in slave arm.

2.2 ROLE OF MECHANICAL ENGINEERING

The field of mechanics plays a very crucial part in the project. The signals received by the controller are used to control the arm. The design of the arm is thus very important.

The arm provides 7 DOF i.e it can copy most of the human arm movements easily including holding and grasping.

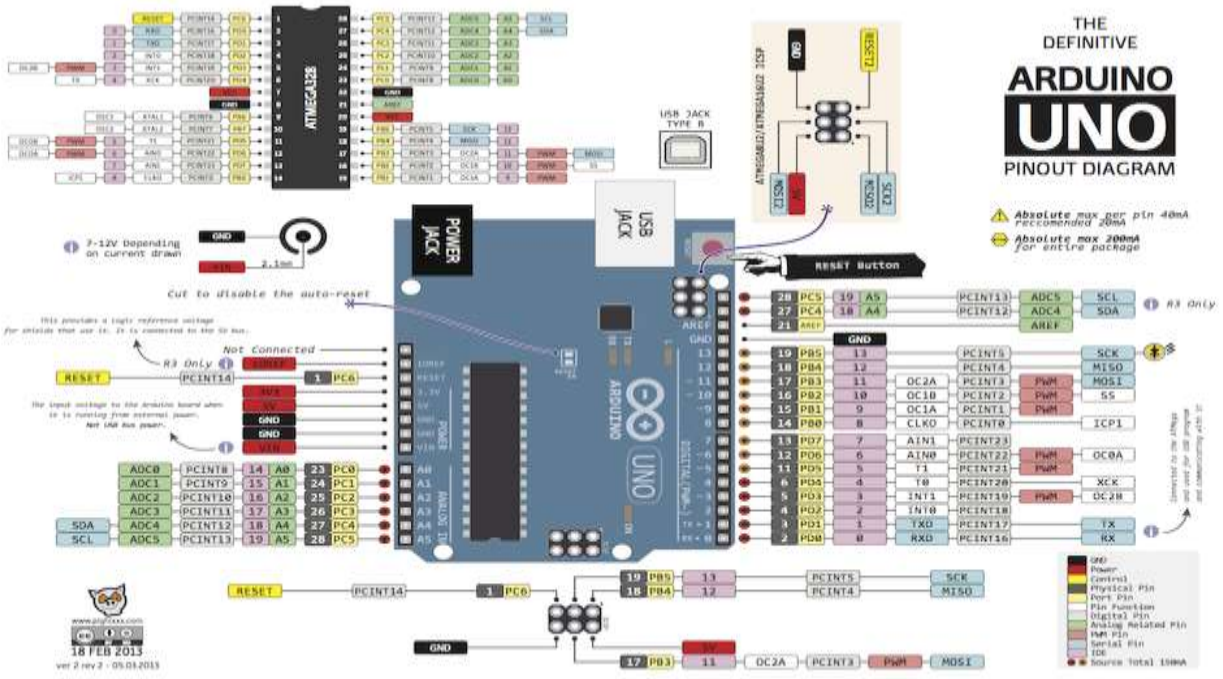
The mechanical engineering plays a very crucial part in selection of motor and chassis designing of the mobile vehicle. We had to choose the motors for both, vehicle and the arms based on different torque requirements and various calculations for it had to be performed. Various comparisons between different types of motors were done to finally decide the servo motor and DC motors which would be used in our project.

And the arm was designed using a 4-bar linkage to produce clamping motion. The pressure of the clamp is derived from a Hitech HS-311 servo motor. The servomotor produces up to a ±90° rotation based on the programming of the microcontroller. Inside the servo motor is a simple potentiometer. The potentiometer, or variable resistor, will allow the motor to rotate until the intended position is reached. Once the position is reached the motor will stop rotating. It will only be prompted to move again if a new signal is provided by the microcontroller.

CHAPTER 3: METHODOLOGY

3.1. ROBOTIC ARMS

3.1.1 ARDUINO



Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based

on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing,MaxMSP).

The boards can be built by hand or purchased preassembled; the software can be downloaded for free. The hardware reference designs (CAD files) are available under an open-source license, you are free to adapt them to your needs.

Arduino received an Honorary Mention in the Digital Communities section of the 2006 Ars Electronica Prix. The Arduino founders are: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis. Credits.

ARDUINO ATmega

- This is a microcontroller board based on the ATmega328 (datasheet).
- It has 14 digital input/output pins (of which 6 can be used as PWM outputs)
- 6 analog inputs.
- A 16 MHz ceramic resonator
- USB connection
- Power jack
- ICSP header
- reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground.

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards

SUMMARY

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)

Schematic: arduino-uno-Rev3-schematic.pdf

Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors

POWER

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.

- IOREF. This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

MEMORY

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

INPUT AND OUTPUT

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), anddigitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labelled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

COMMUNICATION

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

PROGRAMMING

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference,C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

AUTOMATIC (SOFTWARE) RESET

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB OVERCURRENT PROTECTION

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

PHYSICAL CHARACTERISTICS

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

DESIGNING THE ARM

In manipulator structures, stiffness-to-weight ratio of a link is very important since inertia forces induce the largest deflections. Therefore, an increase in the Elastic modulus, E would be very desirable if it is not accompanied by an unacceptable increase in specific density, γ. The Elastic modulus is an indication of the material's resistance to breakage when subjected to force. The best properties are demonstrated by ceramics and beryllium but ceramics have a problem of brittleness and beryllium is very expensive. Structural materials such as magnesium (Mg), aluminium (Al), and titanium (Ti) which are light have about the same E/γ ratios as steel and are used when high strength and low weight are more important than E/γ ratios. Factors like ageing, creep in under constant loads, high thermal expansion coefficient, difficulty in joining with metal parts, high cost and the fact that they are not yet commercially available make the use of fibre-reinforced materials limited though they have good stiffness-to-weight ratios. However, with advances in research, some of the mentioned setbacks have been significantly reduced.. Aluminumlithium alloy have better processing properties and is not very expensive. Alloyedmaterials such as Nitinol (nickel – titanium – aluminium), aluminium incramute (copper -manganese – aluminium) are also commercially available. Therefore the materials recommended for use in this project are

•Al-Li alloys

• Nitinol (nickel-titanium-aluminium)

•Incramute (copper-manganese-aluminum)

•Glass-reinforced Plastic (GRP)The links have an internal hollow area, which provides conduits for power transmitting components i.e. gears in this case, and the stepper motors. At the same time, their external dimensions are limited in order to reduce waste of the usable workspace. They are as light as possible to reduce inertia forces and allow for the highest external load per given size of motors and actuators. For a given weight, links have to possess the highest possible.

Bending (and torsional) stiffness. The parameter to be modified to comply with these constraints is the shape of the cross-section. The choice is between hollow round and hollow rectangular cross-section. From design standpoint of view, the links of square or rectangular cross-section have advantage of strength and machinability ease over round sections. Despite the recommendations mentioned above as regards choice of materials, our options were narrowed down to a choice between steel, GRP, and aluminium based on feasibility studies carried out. Current trend in robotics (especially industrial robotics) shows a quest to achieve lighter designs with reasonable strength. This design goal has always meant a trade-off in terms of cost. Composite materials are generally more expensive than most metals used in industrial robots fabrication. For the particular case of our project, we narrowed our options down to composite material – glass reinforced plastic – otherwise known as GRP and aluminium. The original project, upon which we are building, was fabricated with steel sheets. The sheets cost practically nothing because metal scraps were used, but there was a setback of the motors not being able to cope with the weight of the metal .We figured out at least three ways of overcoming the setback mentioned above. One option would have been to redesign the gear trains and increase torque amplification, so that the motors can support the load. The torque amplification here would have been limited by the real estate on the arm for the gear train and the maximum speed we would be able to give to the motors, as output speed would reduce with increase in torque. We discarded this idea based on long-term considerations. This would mean that much of the energy expended by the robot would go to lifting its own weight thereby reducing the effective load it can lift.

Another option would have been to replace the motors with others having higher torque ratings. This, for us, would almost be as expensive as re-fabricating the arm with a lighter material, and the problem of effective load, as mentioned previously, would still be there. A third option was to re-fabricate the arm, or at least part of it, with a lighter material of reasonable strength, and that was the option we went for. It certainly involved increased short-term costs but then we foresaw a pay off in the long term. We would no longer be constrained to jeopardize the speed or maximum effective load of the robot while trying to increase torque; instead, any torque amplification would directly translate to increased effective load the robot can lift.After more research and consultations with our supervisor and some lecturers in the Mechanical Engineering department, who are experts in the field, we settled for aluminium mainly on grounds of cost and workability.

THE GEAR SYSTEM

In this work, we have chosen the bevel, spur and spiral types of gears. These were readily available from scrap machines (photocopiers). Spiral gears have the advantage of high torque amplification within a relatively small space. The necessary data for the selection and choice of the gear arrangements at each joint

i.Power transmitted, P = TW

ii.Transmitted speed, ω (rad/s)

iii.Torque developed , T (Nm)

iv.Lewis form factor, Y

v.Bending stress, σ ,(ultimate tensile strength)

vi.Ultimate tensile strength

vii.Factor of safety, n

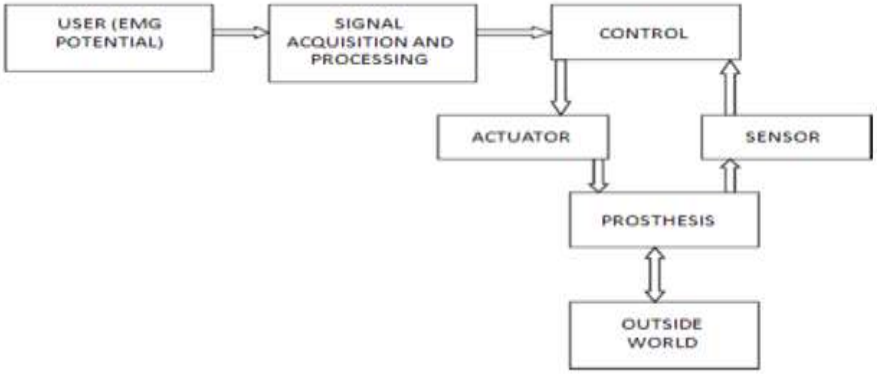
viii.Module of gear, m

ix.Number of teeth, N

CHAPTER:4

EXPERIMENT WORK

4.1 BLOCK DIAGRAM OF EMG HAND



4.1.2 MICROCONTROLLER

The EMG signals are fed to the microcontroller. According to the voltage levels, the Servo motor is controlled. The controller converts the analog signals into the PWM(Pulse Width Modulated)signal for driving the motor.

4.1.3 SERVO MOTOR

The signals sent to the motor control and rotate it in the clockwise or anti-clockwise direction. The servo motor that is being used has the drivers and negative feedback position control loop hard- wired.

4.4 SERVO LIBRARY

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

4.4.1 CIRCUIT CONNECTIONS

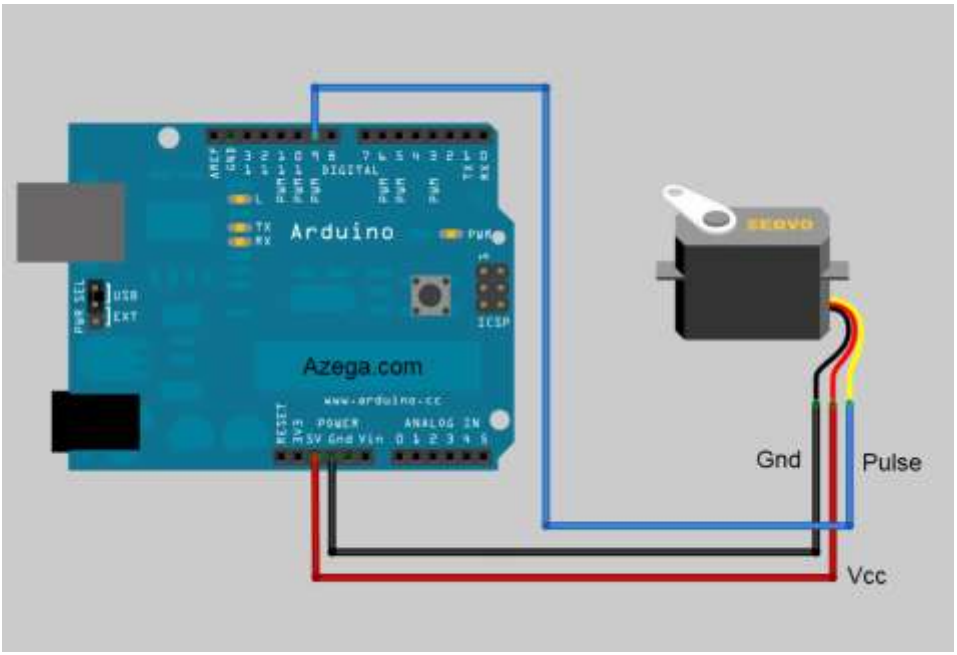


Fig: Interfacing Servo Motor with Arduino

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow, orange or white and should be connected to a digital pin on the Arduino board. Note that servos draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply (i.e. not the +5V pin on your Arduino). Be sure to connect the grounds of the Arduino and external power supply together.

4.4.2 FUNCTIONS USED

4.4.2.1 attach ()

- Description

Attach the Servo variable to a pin. Note that in Arduino 0016 and earlier, the Servo library supports only servos on only two pins: 9 and 10.

- Syntax

servo.attach(pin)

servo.attach(pin, min, max)

- Parameters

servo: a variable of type Servo

pin: the number of the pin that the servo is attached to

min (optional): the pulse width, in microseconds, corresponding to the minimum (0-degree) angle on the servo (defaults to 544)

max (optional): the pulse width, in microseconds, corresponding to the maximum (180-degree) angle on the servo (defaults to 2400).

4.4.2.2 write()

- Description

Writes a value to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation. On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).

- Syntax

servo.write(angle)

- Parameters

servo: a variable of type Servo

angle: the value to write to the servo, from 0 to 180

4.4.2.3 pinMode()

- Description

Configures the specified pin to behave either as an input or an output. See the description of digital pins for details on the functionality of the pins.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

- Syntax

pinMode(pin, mode)

- Parameters

pin: the number of the pin whose mode you wish to set

mode: INPUT, OUTPUT, or INPUT_PULLUP. (see the digital pins page for a more complete description of the functionality.)

- Returns

None

4.4.2.4 analogRead()

- Description

Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

- Syntax

analogRead(pin)

- Parameters

pin: the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

- Returns

int (0 to 1023)

- Note

If the analog input pin is not connected to anything, the value returned by analogRead() will fluctuate based on a number of factors (e.g. the values of the other analog inputs, how close your hand is to the board, etc.

4.4.3 CODE FOR READING ANALOG INPUT AND CONTROL THE SERVO MOTOR

```
#include <Servo.h>// include the servo library
Servo myServo;  // create a servo object
int const potPin = A0; // analog pin used to connect the  circuit
int potVal;  // variable to read the value from the analog pin
//int angle;   // variable to hold the angle for the servo motor
int pos = 0;    // variable to store the servo position
void setup()
{
  pos = 0; /* Initialise pos */
 myServo.write(0);    /*When you create a new Servo object, its position
                        is automatically given a default of 90. By setting a position first,
                         Then attaching the servo, you can have it start at a
                          position other than 90.*/
 myServo.attach(9);   // attaches the servo on pin 9 to the servo object
```

```
 pinMode(potVal, INPUT);
}



void loop()
 {
 potVal = analogRead(potPin); // read the value of the potentiometer
 // scale the numbers from the pot
if(potVal>=204&& potVal <=446)// if analog voltage is between 0.5 v to
                                        2.18volts then move the motor clockwise.


for(pos; pos <= 170; pos += 1)  /* pos need not be at 0 for this to work.*
 {
   myServo.write(pos);              // tell servo to go to position  to variable 'pos'
   delay(15);                                 // waits 15ms for the servo to reach
                                     //the position.
 }
 }
 else
 {
for(pos; pos >= 0; pos -= 1) /* pos need not be at 170 for this to work.
                                     If potVal becomes < 204 rotate servo anticlockwise
{                                     in steps of 1 degree*/
   myServo.write(pos);              // tell servo to go to position in
                                     //  variable 'pos'
   delay(15);                    // waits 15ms for the servo to reach
```

```
                    //  the position

}
```

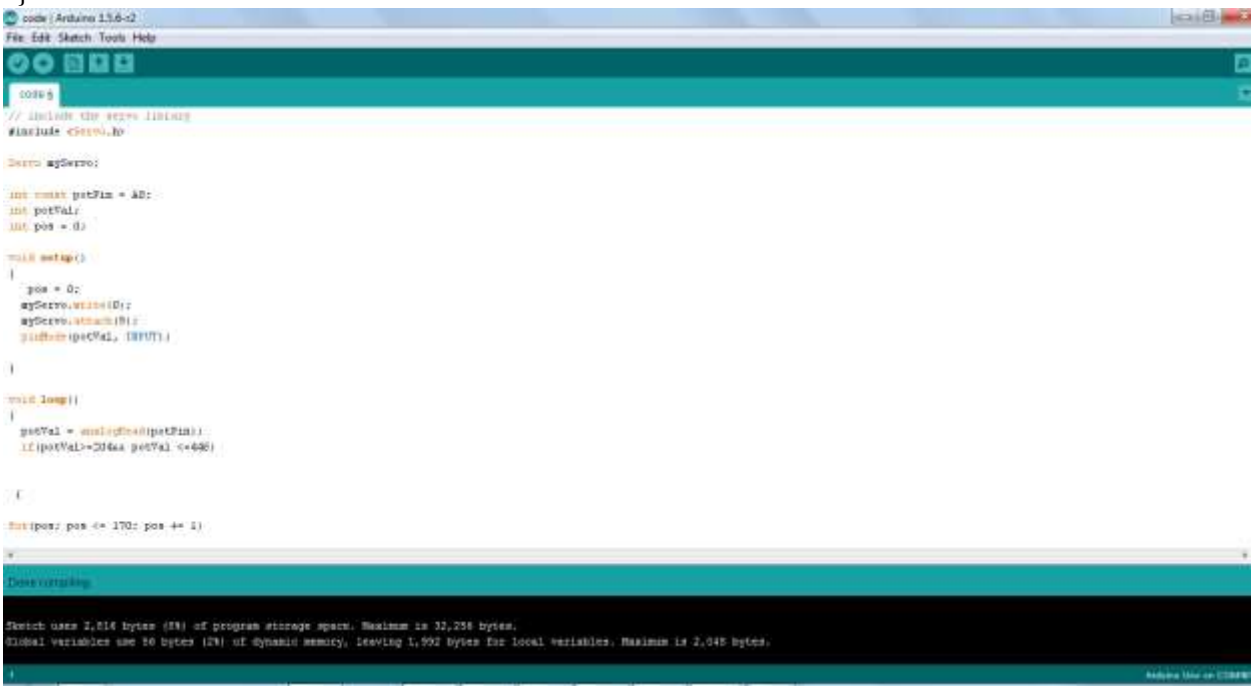


Fig :Screenshot of upper half of code

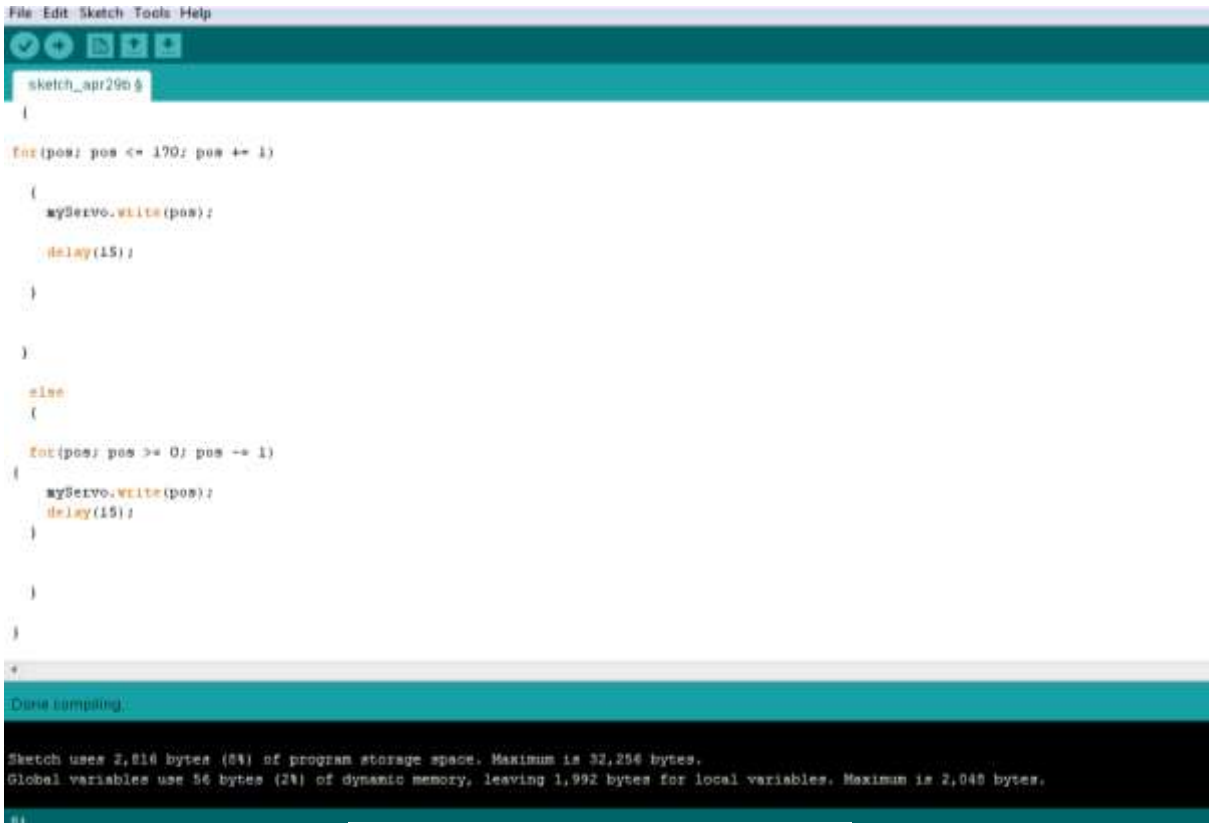


Fig: Screenshot of lower half of code

A rough estimate of the total cost of the prototype has been made in the table below:

Sr. No.	Component	Number of components	Cost per component(INR)	Total cost(INR)
1	DC Servo Motors	4	750	3000
2	Microcontroller (Arduino)	1	1500	1500
3	Integrated circuitry: Op amp LF-351	9	15	135
4	Integrated circuitry: INA 128p	3	450	1350
5	Surface electrodes	7	20	140
6	Mechanical components: Finger materials	3	-	-
			TOTAL	6125(approx)

REFERENCES:

PAPERS:

- Mechatronic Experiments Course Design: A Myoelectric Controlled Partial-Hand Prosthesis Project
- Design and Implementation of Prosthetic Arm Using Gear Motor Control Technique With Appropriate Testing

SITES:

- http://arxiv.org/ftp/arxiv/papers/1111/1111.2258.pdf
- http://www.scribd.com/doc/18651364/Myoelectric-Arm
- http://www.g9toengineering.com/MechatronicsStudio/MyoelectricProsthesis.htm
- http://i4d.mit.edu/prosthetic-arm/