

Robotic Arm Using Arduino Microcontroller and DC Motors (2014)

- Soham Parekh
Siddhath Thevaril
Ashish Sharma



Laxmi Singh Charitable Trust's (a/k/a)

**THAKUR COLLEGE OF
ENGINEERING & TECHNOLOGY**

(Approved by AICTE, Govt. of Maharashtra & Affiliated to University of Mumbai)
(Accredited by National Board of Accreditation, New Delhi)*

* Accredited Programmes : • Computer Engineering • Electronics & Telecommunication Engineering • Information Technology (w.e.f. 16-09-2011 for 3 years)

A - Block, Thakur Educational Campus,
Shyamnarayan Thakur Marg, Thakur Village,
Kandivoli (East), Mumbai - 400 101.
Tel.: 6730 8000 / 8106 / 8107
Fax : 2846 1890
Email : tcet@thakureducation.org
Website : www.tcetmumbai.in • www.thakureducation.org



CONTENTS

Chapter No.	Topic	Pg. No.
	• List of figure	i
	• List of tables	ii
	• Abbreviation and symbols	iii
	• Definitions	iv
Chapter 1	Introduction	
	1.1 Importance of the Project and its background	1
	1.2 Literature survey	1
	1.3 Motivation	3
	1.3 Scope of the project	4
	1.5 Organization of the Project report	
Chapter 2	Proposed Work	
	2.1 Problem Definition	6
	2.2 Flow chart of Design	7
Chapter 3	Analysis and Planning	
	3.1 Feasibility Study	8
	3.2 Project Planning	8
	3.3 Scheduling (Time line chart)	11
Chapter 4	Design and Implementation	
	4.1 Technology/ Software	12
	4.2 Installation stage	27
Chapter 5	Results and Discussion	
	Results and discussion for algorithm 1	28
Chapter 6	Conclusion and Scope for future work	29
	Appendix	
	References	



Laxda Singh Charitable Trust's (Regd.)

**THAKUR COLLEGE OF
ENGINEERING & TECHNOLOGY**

(Approved by AICTE, Govt. of Maharashtra & Affiliated to University of Mumbai)

(Accredited by National Board of Accreditation, New Delhi)

* Accredited Programmes : • Computer Engineering • Electronics & Telecommunication Engineering • Information Technology (w.e.f.: 16-09-2011 for 3 years)

A - Block, Thakur Educational Campus,
Shyamnarayan Thakur Marg, Thakur Village,
Kandivali (East), Mumbai - 400 101.

Tel.: 6730 8000 / 8106 / 8107

Fax : 2846 1890

Email : tcet@thakureducation.org

Website : www.tcetmumbai.in • www.thakureducation.org



List of Tables

Fig No.	Name of the table	
3.2	Planning Table	10
3.3	Time Line Chart	11
4.1.1	Arduino Specifications	14
4.1.5	Hardware and software specifications	22



Laxda Singh Charitable Trust's (Regd.)
**THAKUR COLLEGE OF
ENGINEERING & TECHNOLOGY**
(Approved by AICTE, Govt. of Maharashtra & Affiliated to University of Mumbai)
(Accredited by National Board of Accreditation, New Delhi)
* Accredited Programmes : • Computer Engineering • Electronics & Telecommunication Engineering • Information Technology (w.e.f.: 16-09-2011 for 3 years)

A - Block, Thakur Educational Campus,
Shyamnarayan Thakur Marg, Thakur Village,
Kandivali (East), Mumbai - 400 101.
Tel.: 6730 8000 / 8106 / 8107
Fax : 2846 1890
Email : tcet@thakureducation.org
Website : www.tcetmumbai.in • www.thakureducation.org



ABBREVIATIONS

Vcc – Voltage Supply

DoF – Degree of Freedom

GND – Ground

IDE – Integrated Development Environment



Laxmi Singh Charitable Trust's (Regd.)

**THAKUR COLLEGE OF
ENGINEERING & TECHNOLOGY**

(Approved by AICTE, Govt. of Maharashtra & Affiliated to University of Mumbai)
(Accredited by National Board of Accreditation, New Delhi)*

* Accredited Programmes : • Computer Engineering • Electronics & Telecommunication Engineering • Information Technology (w.e.f.: 16-09-2011 for 3 years)

A - Block, Thakur Educational Campus,
Shyamnarayan Thakur Marg, Thakur Village,
Kandivali (East), Mumbai - 400 101.

Tel.: 6730 8000 / 8106 / 8107

Fax : 2846 1890

Email : tcet@thakureducation.org

Website : www.tcetmumbai.in • www.thakureducation.org



DEFINITIONS

DC: DC stands for Direct Current.

Robotics: It is a branch of that involves conception, design and manufacture of robots.

End effector: Connects to the robot's arm and functions as a hand. This part comes in direct contact with the material the robot is manipulating.

Actuator: An actuator is a type of motor for moving or controlling a mechanism or system. It is operated by a source of energy, typically electric current, hydraulic fluid.

CHAPTER 1: INTRODUCTION

1.1. Importance of the Project and its background

Robots are not just machines, they are many steps ahead a typical machine. Robots like machines can perform different tough jobs easily but the advancement is that they can do it by their own. Once programmed robots can perform required tasks repeatedly in exactly the same way.

The modern definition of a robot can be an electro-mechanical device which follows a set of instructions to carry out certain jobs, but literally robot means a „slave“. Robots find wide application in industries and thus are called there as industrial robots and also in sci-fi movies as humanoids. This and coming articles will provide an introduction to the Robotics. When we think about robotics first thing that come to our mind is automation. Robots are known to perform tasks automatically without much human intervention, except for initial programming and instruction set being provided to them. The first machine, what I have seen in my childhood when we were on a visit to a milk processing plant, most close, to be called as a robot was a milk packaging machine. There was roll of packaging material running through the machine, each time half a liter of milk falls into the roll and then a mechanism in the machine seals and cuts the packet.

This machine can be a simple example of a very basic robot. It performs the specified sequence of operations repeatedly with the same accuracy. It was programmed and provided with the required material and then started.

1.2. Literature Survey

1. 5-DOF PC-Based Robotic Arm (PC-ROBOARM) with efficient trajectory planning and speed control, Published in Mechatronics (ICOM), 2011 4th International Conference On 19th May 2011

This paper introduces the design and development of 5-DOF (degree of freedom) PCBased Robotic Arm (PC-ROBOARM). The main context of the study is concerning a 5-DOF robotic arm, which is modeled as three-link, with each joint connected with a suitable servomotor. The robotic arm design and control solution is implemented by self-developed computer software which is named as SMART ARM. It is a computer aided design and control solution for 5-DOF robotic arm which come with an user friendly graphical user interface

(GUI). It allows user to model or design virtual robotic arm before building the real one. Therefore, the user can estimate the optimum size of actual robotic arm at the beginning so as to minimize the building cost and suite the practical environment. Furthermore, once the actual robotic arm has been built, the user can reuse the software to control the actual robotic arm in an effortless way without wasting time in constructing new control solution. The software also provides simulation feature. Through simulation in the GUI, the software assists greatly in visualizing the robotic arm trajectory planning.

2. Motion planning and control of interactive humanoid robotic arms, Published in:

Advanced robotics and Its Social Impacts, 2008. ARSO 2008. IEEE Workshop

Date of Conference: 23-25 Aug. 2008

In this paper, we propose a robotic arm which manipulation is analog to the motion of humans upper extremities. The proposed robotic arm is designed as a seven degree-of-freedom configuration. To increase the interactivity with humans, a six-axis force sensor is attached on the wrist of the robot to capture the force applied on the robotic arm. Subsequently, the robotic arm is moved following the force applied on the wrist. In addition to the compliance of humans motion, the robotic arm is capable of dynamically planning spatial trajectories for various straight lines, circles, or predefined paths. Especially, due to the structure of this seven degree-of-freedom robotic arm, we cannot find a unique solution for the inverse kinematics. In this work, we present a behavior based inverse kinematics approach to solve this problem in terms of the fuzzy reasoning. Various behaviors for a given spatial position or path, such as writing, pickup, etc., may result different inverse kinematic solution, and may generate different elbow trajectories as well. Therefore, the proposed robotic arm not only has similar structure to humans, but also represents similar behavior to humans. More specially, the compliance function makes this robotic arm possible to interact with humans. Consequently, a robotic arm with tendon driven architecture is demonstrated to validate the proposed motion planning and control approaches based on an ARM based controller.

3. The multiple-function intelligent robotic arm, Published in:
Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on

Date of Conference: 20-24 Aug. 2009

This paper details the design, production, and programming methodology of a multiplefunction robotic arm. All of the hardware and software of this robotic arm were designed and produced by the authors. This robotic arm won the championship of the first competition of HIWIN intelligent robotic arms on Aug. 22, 2008 in Taiwan. The main design goal of this robotic arm was to present the following functions: fancy dancing, weight lifting, Chinese calligraphy, and color classification. Another design goal was to minimize cost and maximize performance. On the other hand, a set of the robotic arms are also applied to show martial arts and play the rock-scissors-paper game with modifying and mimetic hands. The characteristics of the robotic arms include: 1. The shoulder of robotic arm includes a pair of motor structures to enhance the ability to life weight. 2. The stability and accurateness of the robotic arm are optimized for the requirement of high performance throughout the whole structural design. 3. In order to increase the moving ability of the robot, the robotic arm was designed with a fourwheeled transmission structure and track. 4. Two mimetic robotic arms work in concert to present the fancy shows. 5. Five kinds of machine hands were designed to meet the requirements of the six appointed functions.

1.3. Motivation

The purpose of this project is to create a Robotic Arm whose primary Goal is to pick and place objects. This was inspired from the movie „Iron Man“ in which a robotic arm called J.A.R.V.I.S. (Just A Rather Very Intelligent System) did a similar job. J.A.R.V.I.S. is Tony Stark's (Protagonist) home computing system, taking care of everything to do with the house, from Heating and Cooling Systems to Engine Analysis of Stark's hot rod in the garage.

J.A.R.V.I.S. Although the movie was a fiction and it's not possible to create the same machine. However we decided to implement the pick-and-place mechanism.

The arm will be controlled by a microcontroller called „Arduino UNO“ which will be the „brain“ of the arm. The advantage of using the Arduino is that it simplifies the process of working with microcontrollers. It offers some advantage for teachers, students, and interested amateurs over other systems. In this way we can save time by skipping the redundant aspects and concentrating on the useful logic.

This project work would also be useful to those who study Robotics and would want to add more features to create a more refined product.

Finally, we hope that our work will allow us learn more about the microcontroller and working of the system and also exploring the possibilities to create a better system.

1.4. Scope of the project

Robotics engineers can design robots which can do a whole lot of things, ranging from delicate and precision tasks such as fitting small parts of watches and to the hazardous tasks such as fuelling the chambers of nuclear reactors. Robots are thought to be super-machines but they have limitations.

Despite the great advancements in the field of robotics and continuous efforts to make robots more and more sophisticated to match the capabilities of human beings and even surpass them, still, from a very scientific and logical point of view, robots developed up till these days are no way closer to human beings.

In basic robotics we design machines to do the specified tasks and in the advanced version of it robots are designed to be adaptive, that is, respond according to the changing environment and even autonomous, that is, capable to make decisions on their own. While designing a robot the most important thing to be taken in consideration is, obviously, the function to be performed. Here comes into play the discussion about the scope of the robot and robotics. Robots have basic levels of complexity and each level has its scope for performing the requisite function.

The levels of complexity of robots is defined by the members used in its limbs, number of limbs, number of actuators and sensors used and for advanced robots the type and number of microprocessors and microcontrollers used. Each increasing component adds to the scope of functionality of a robot. With every joint added, the degrees of freedom in which a robot can work increases and with the quality of the microprocessors and microcontrollers the accuracy and effectiveness with which a robot can work is enhanced.

1.5. Organization of the Project report

The report begins with Chapter-1 : Introduction to the world of robotics, its scope, applications and its importance as an automated system.

Chapter-2 : Proposed Work and Literature Review explains about the functional properties and characteristics of the 5-DOF Robot arm and provides links various resources that were referenced in the development of the arm. It gives an insight about the working of the Robot arm through a flow chart diagram.

Chapter-3 Analysis and Planning represents the planning schedule of the entire project. The entire phase is represented with the help of a schedule chart, or otherwise called a Gantt chart.

Chapter-4 deals with the technology used to construct the project. This includes the various peripheral components such as the microcontroller and photo-detector. The schematic diagram of the entire physical layout is described in this chapter.

In basic robotics we design machines to do the specified tasks and in the advanced version of it robots are designed to be adaptive, that is, respond according to the changing environment and even autonomous, that is, capable to make decisions on their own. While designing a robot the most important thing to be taken in consideration is, obviously, the function to be performed. Here comes into play the discussion about the scope of the robot and robotics. Robots have basic levels of complexity and each level has its scope for performing the requisite function.

CHAPTER-2: PROPOSED WORK AND LITERATURE REVIEW

2.1. Problem Definition

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm; the arm may be the sum total of the mechanism or may be part of a more complex robot. The links of such a manipulator are connected by joints allowing either rotational motion (such as in an articulated robot) or translational (linear) displacement. The links of the manipulator can be considered to form a kinematic chain. The terminus of the kinematic chain of the manipulator is called the end effector and it is analogous to the human hand.

The end effector, or robotic hand, can be designed to perform any desired task such as welding, gripping, spinning etc., depending on the application. For example robot arms in automotive assembly lines perform a variety of tasks such as welding and parts rotation and placement during assembly. In some circumstances, close emulation of the human hand is desired, as in robots designed to conduct bomb disarmament and disposal.

The Physical layout of the system is shown in the diagram below. As we can see the various terminals of the Arduino are connected to the required terminals of the bridge. The Arduino will provide low level input to the bridge i.e. all the pins of the microcontroller will be assigned as „Output“ mode. The H-bridge will provide the voltage levels required for the motors to actuate.

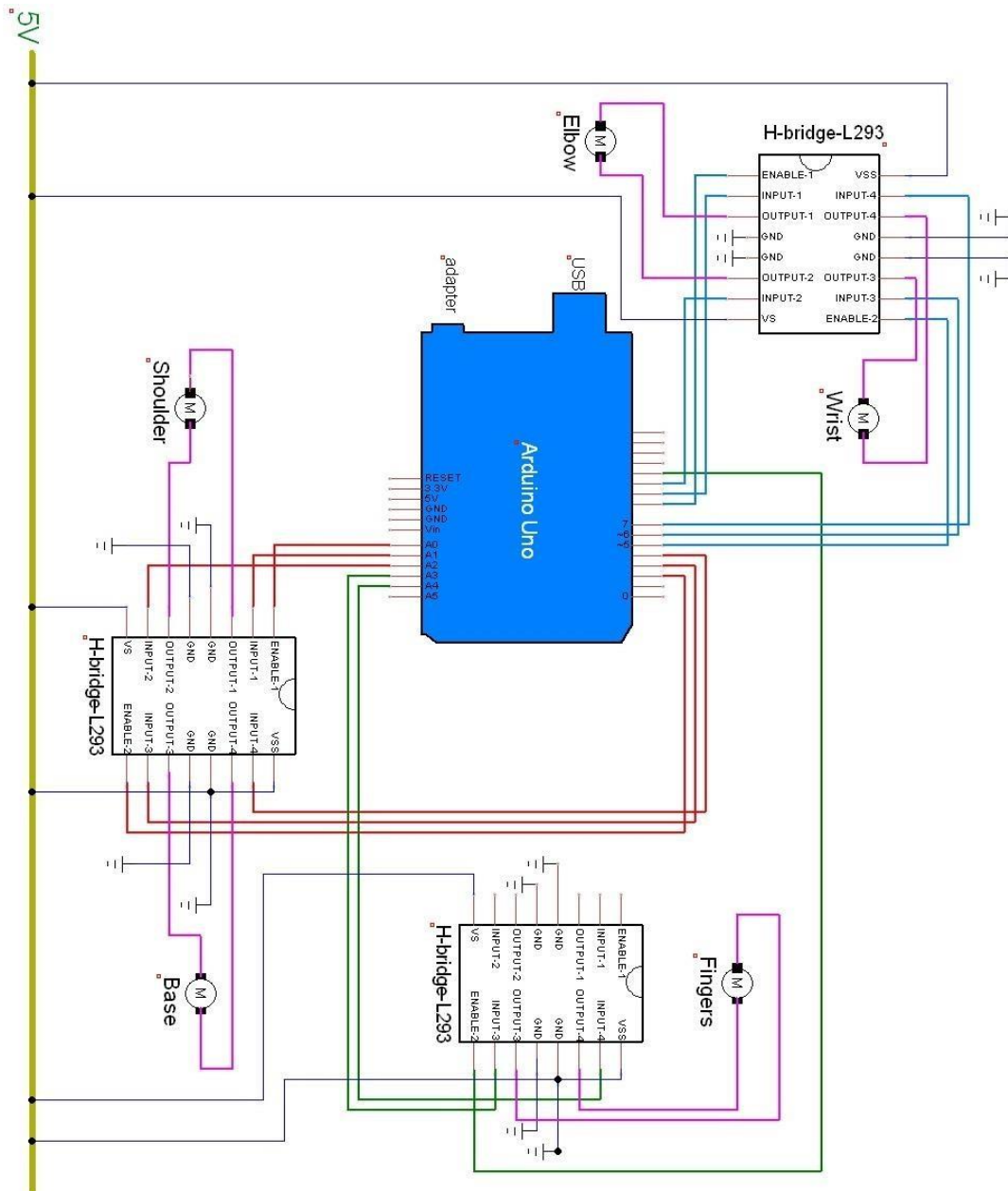


Fig 2.1: System architecture of Arduino micro controller based Robot Arm.

2.2. Flow Chart

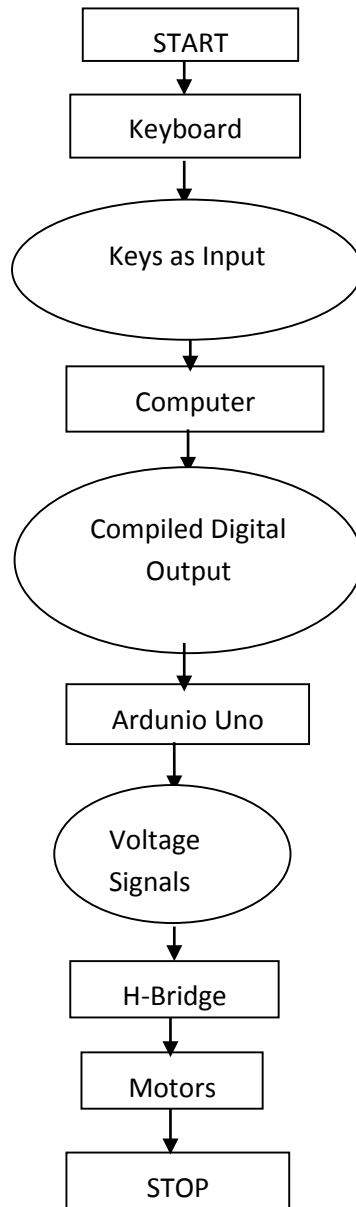


Fig 2.1: Data Flow Diagram of the Robotic Arm

CHAPTER-3: ANALYSIS AND PLANNING MANAGEMENT

3.1. Feasibility Study

The proposed project is to pick up the object from the source and place it at a desired destination. This would require object (maximum of 10 grams). The input will be given manually by the user by using a keyboard for the moment of the arm and it will be supplied Arduino UNO which will perform the required computation and will move arm. It has a keyboard to supply the input, Arduino UNO microcontroller, H-bridge to provide output to the motors to move the arm. After an analysis of research papers on the subject, a rough algorithm is also developed that enables implementation of the topic. The implementation is to be done using C/C++. The software requirements would require a minimum of Windows XP and Arduino IDE setup. The hardware needed would be at least 512 MB RAM, Pentium IV, 40 GB hard disk, keyboard and a monitor. These are easily met and hence the project is concluded to be feasible.

3.2. Project Planning

3.2.1 Development of Project Topic

The project topic was decided in June and was tentative. The topic was finalized by the end of July with the help of the project guide and project coordinator after properly examining the content.

Duration: Around 2 weeks

3.2.2 Tracking the Project

Analysis was done on the topic and content was gathered. Different research papers based on this were also examined.

Duration: Around 3 weeks

3.3.3 Preliminary Design

The preliminary design included an overview of how the system would look like. Logical flow diagrams like DFD's were created.

Duration: Around 1 week

3.3.4 Detailed Design

Detailed design included system architecture and specifications of various components used. It demonstrated the logical connections of various terminals to their desired destination terminals. **Duration:** Around 1 week

3.3.5 Document Design

The Document design included gathering of the project specification and documenting them together. This includes the abstract of the proposed work, literature review etc.

Duration: Around 1.5 weeks

3.3.6 Review Design

The design of the project was reviewed with the help of the project guide.

Duration: Around 1 week

3.3.7 Integrating the hardware modules

The physical components were procured. The modules were developed and then integrated by connection of the required modules to their appropriate terminals. A Physical System was thus formed.

Duration: Around 3 weeks

3.3.8 Development of Software

The IDE for coding of the software was procured. The software required for the functioning of the project was developed by understanding the required functionalities of the system. The appropriate code was thus developed.

Duration: Around 2 weeks

3.3.9 Interfacing the hardware and the software

The hardware and the software were interfaced using a computer system. The code developed was uploaded into the physical system. The Product was thus ready to function.

Duration: Around 2 days

3.3.10 Testing

After the system was ready, testing was done for its overall functionalities. The testing was done in two parts, Hardware testing and Software testing.

Hardware Testing included testing of the physical components like end-effectors, elbow etc. for their functionalities.





Software testing included testing the code and detection flaws like a method missing, invalid method call etc.

3.2 Project planning/Timeline

Sr. No	Task	Activity	Duration	Start Date	End Date
1	Finalizing our project from 3-4 topics	We researched and found the topic which had some drawback and we can enhance it	7 days	16/08/2013	22/08/2013
2	Finalizing the topic with project coordinator.	After researching, we needed the approval of our project coordinator	7 days	23/08/2013	29/08/2013
3	Allocation of Project Guide	In this week, we were allocated Project Guide.	7 days	30/08/2013	05/09/2013
4	Starting our project with documenting the idea	Once our project guide was allocated, we discussed the idea with our project guide	7 days	06/09/2013	12/09/2013

5	Deciding the technology	Checking different technologies and finalizing .net	7 days	13/09/2013	19/09/2013
6	Requirement Gathering	In this phase, we started grabbing the data required for our project	7 days	20/09/2013	26/09/2013
7	Preparing the preliminary report	We stated making the report of our work	7 days	27/09/2013	03/10/2013
8	Finalizing the report	We finally documented our project in a book and submitted it to our project guide.	7 days	04/10/2013	10/10/2013

3.3. Scheduling/Progress (Time line chart)

Sr No.	Activity	NOVEMBER	DECEMBER	JANUARY	FEBRUARY	MARCH
	Dates	1-8 9-16 17-23 24-30	1-8 9-16 17-23 24-31	1-8 9-16 17-23 24-31	1-8 9-16 17-23 24-28	1-8 9-16 17-23 24-31
1	Preliminary investigation					
2	System Analysis					
3	System Design					
4	System					

	Coding					
5	Maintenance & Evaluation					
6	Project					

CHAPTER-4: DESIGN AND IMPLEMENTATION

4.1. Technology/Software

4.1.1 Arduino Uno:

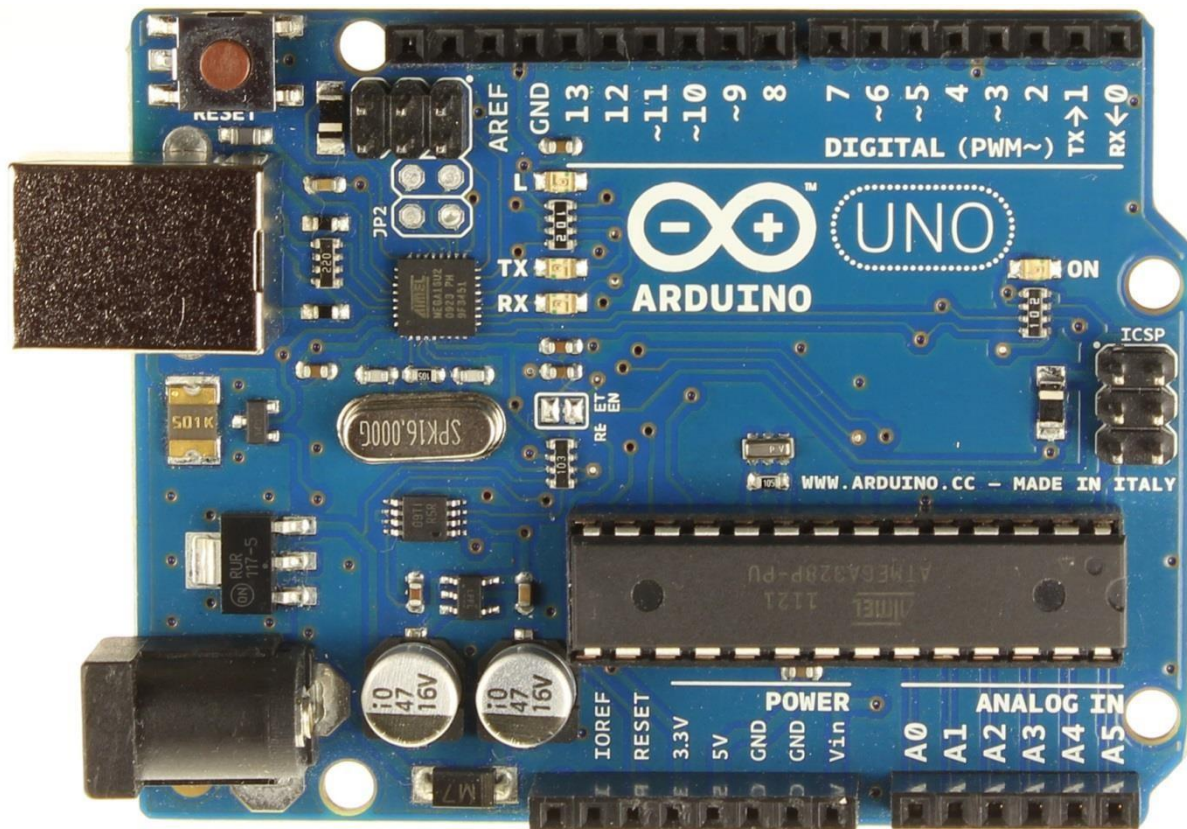


Fig 4.1: Arduino Uno micro-controller

Arduino is a single-board microcontroller to make using electronics in multidisciplinary projects more accessible. The hardware consists of an open-source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.

An Arduino board consists of an Atmel 8-bit AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits.

An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus, allowing many shields to be stacked and used in parallel. Official Arduinos have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants),

although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a level shifter circuit to convert between RS232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. (When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.)

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available.

The Arduino Uno 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, a USB connection, a power jack, an ICSP header, and a 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.

"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

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. We don't advise it.

- **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(), and digitalRead() 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, labeled 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.

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.

4.1.2 H-Bridge L293:

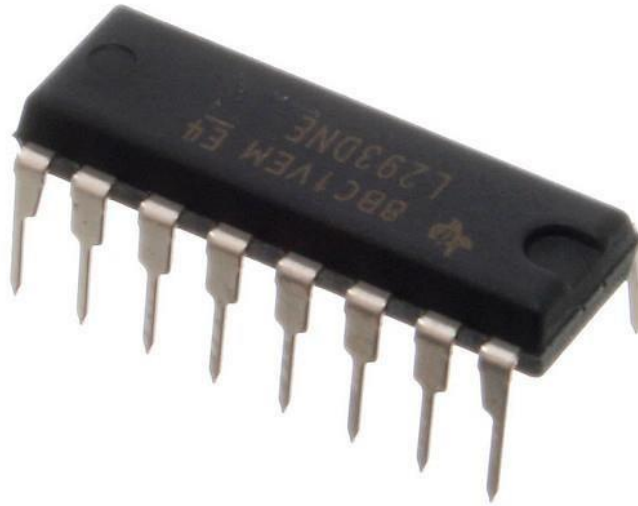


Fig 4.2: H-bridge L293

An H bridge is an electronic circuit that enables a voltage to be applied across a load in either direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards and backwards. Most DC-to-AC converters (power inverters), most AC/AC converters, the DC-to-DC push-pull converter, most motor controllers, and many other kinds of power electronics use H bridges. In particular, a bipolar stepper motor is almost invariably driven by a motor controller containing two H-bridges.

H bridges are available as integrated circuits, or can be built from discrete components. The term *H-bridge* is derived from the typical graphical representation of such a circuit. An H bridge is built with four switches (solid-state or mechanical). When the switches S1 and S4 (according to the first figure) are closed (and S2 and S3 are open) a positive voltage will be applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing reverse operation of the motor.

Using the nomenclature above, the switches S1 and S2 should never be closed at the same time, as this would cause a short circuit on the input voltage source. The same applies to the switches S3 and S4. This condition is known as shoot-through.

CONNECTION DIAGRAMS

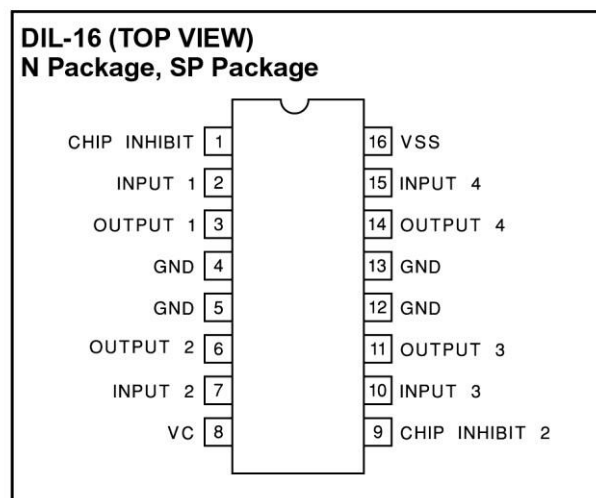


Fig 4.3: H bridge L293 pin diagram

The H-bridge arrangement is generally used to reverse the polarity of the motor, but can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively disconnected from the circuit. The following table summarises operation, with S1-S4 corresponding to the diagram above.

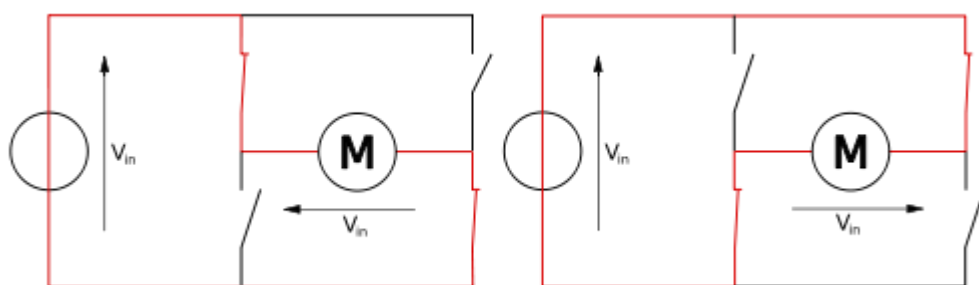


Fig 4.4: Circuit diagram of an H Bridge L293

4.1.3 DC Motor

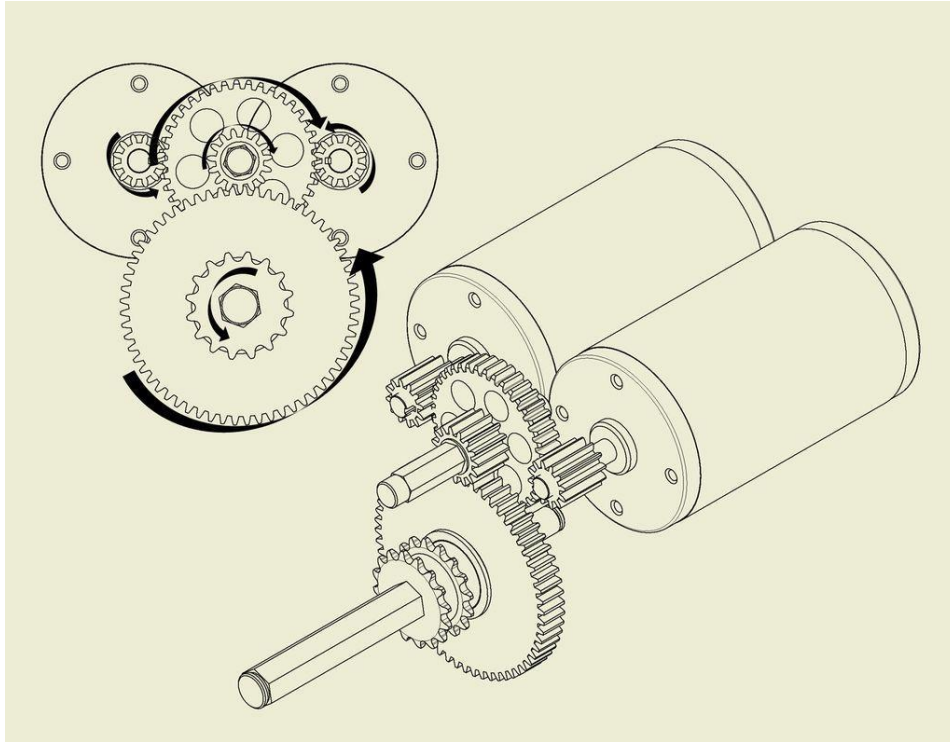


Fig 4.5: DC Motor

A DC motor is a mechanically commutated electric motor powered from direct current (DC). The stator is stationary in space by definition and therefore the current in the rotor is switched by the commutator to also be stationary in space. This is how the relative angle between the stator and rotor magnetic flux is maintained near 90 degrees, which generates the maximum torque.

DC motors have a rotating armature winding (winding in which a voltage is induced) but non-rotating armature magnetic field and a static field winding (winding that produce the main magnetic flux) or permanent magnet. Different connections of the field and armature winding provide different inherent speed/torque regulation characteristics. The speed of a DC motor can be controlled by changing the voltage applied to the armature or by changing the field current. The introduction of variable resistance in the armature circuit or field circuit allowed speed control. Modern DC motors are often controlled by power electronics systems called DC drives.

The introduction of DC motors to run machinery eliminated the need for local steam or internal combustion engines, and line shaft drive systems. DC motors can operate directly from rechargeable batteries, providing the motive power for the first electric vehicles. Today

DC motors are still found in applications as small as toys and disk drives, or in large sizes to operate steel rolling mills and paper machines.

4.1.4 5-DoF Arm

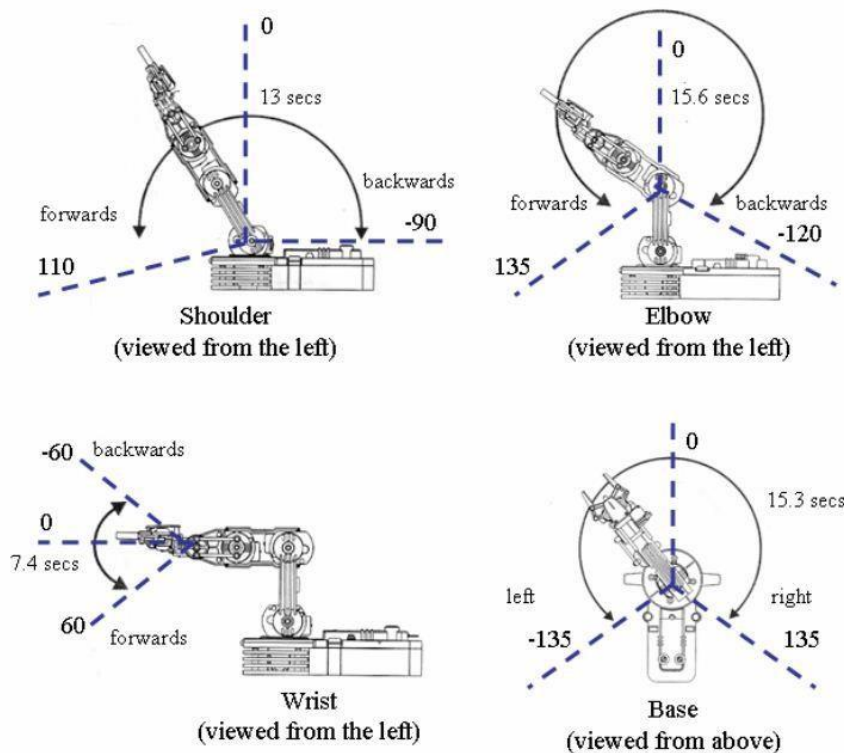


Fig 4.6: Components of a robotic arm

- Weight: 658 g
- Lifting Capacity: 100 g
- 9.0" L x 6.3" W x 15.0" H
- Power Source: 4 D Cell Batteries
- Maximum Vertical Reach: 15"
- Maximum Horizontal Reach: 12.6"
- Wrist Motion Range: 120°
- Elbow Motion Range: 300°

- Base Motion (shoulder) Range: 180°
- Base Rotation Range: 270°

4.1.5 Photo resistor (Light sensor)

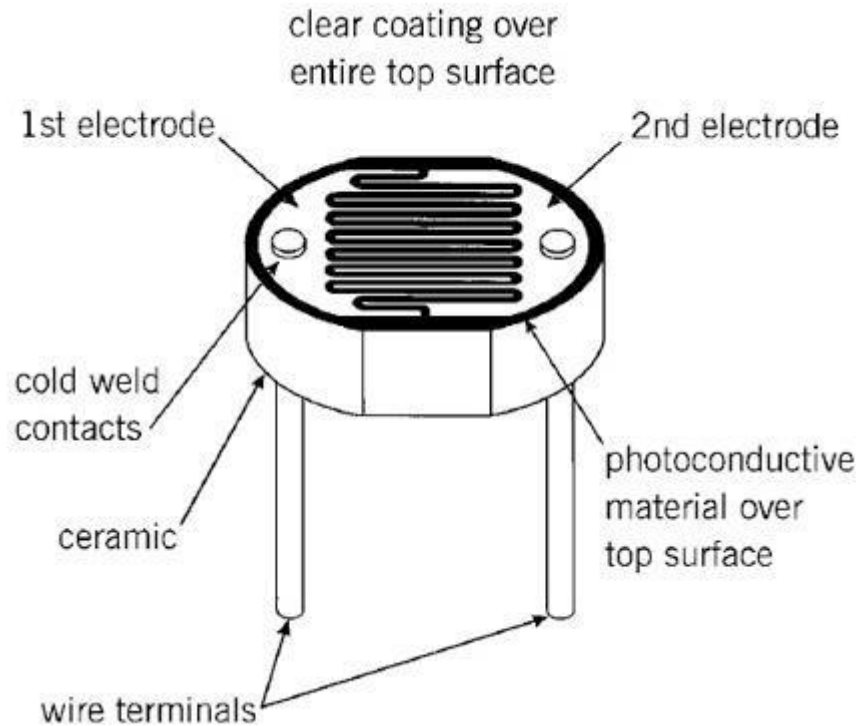


Fig 4.7: Photo resistor light Sensor

A photoresistor or light-dependent resistor (LDR) or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as a few megaohms ($M\Omega$), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photoresistor

can substantially differ among dissimilar devices. Moreover, unique photoresistors may react substantially differently to photons within certain wavelength bands.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, for example, silicon. In intrinsic devices the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electron across the entire bandgap. Extrinsic devices have impurities, also called dopants, added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (that is, longer wavelengths and lower frequencies) are sufficient to trigger the device. If a sample of silicon has some of its atoms replaced by phosphorus atoms (impurities), there will be extra electrons available for conduction. This is an example of an extrinsic semiconductor.

4.1.5 Arduino IDE

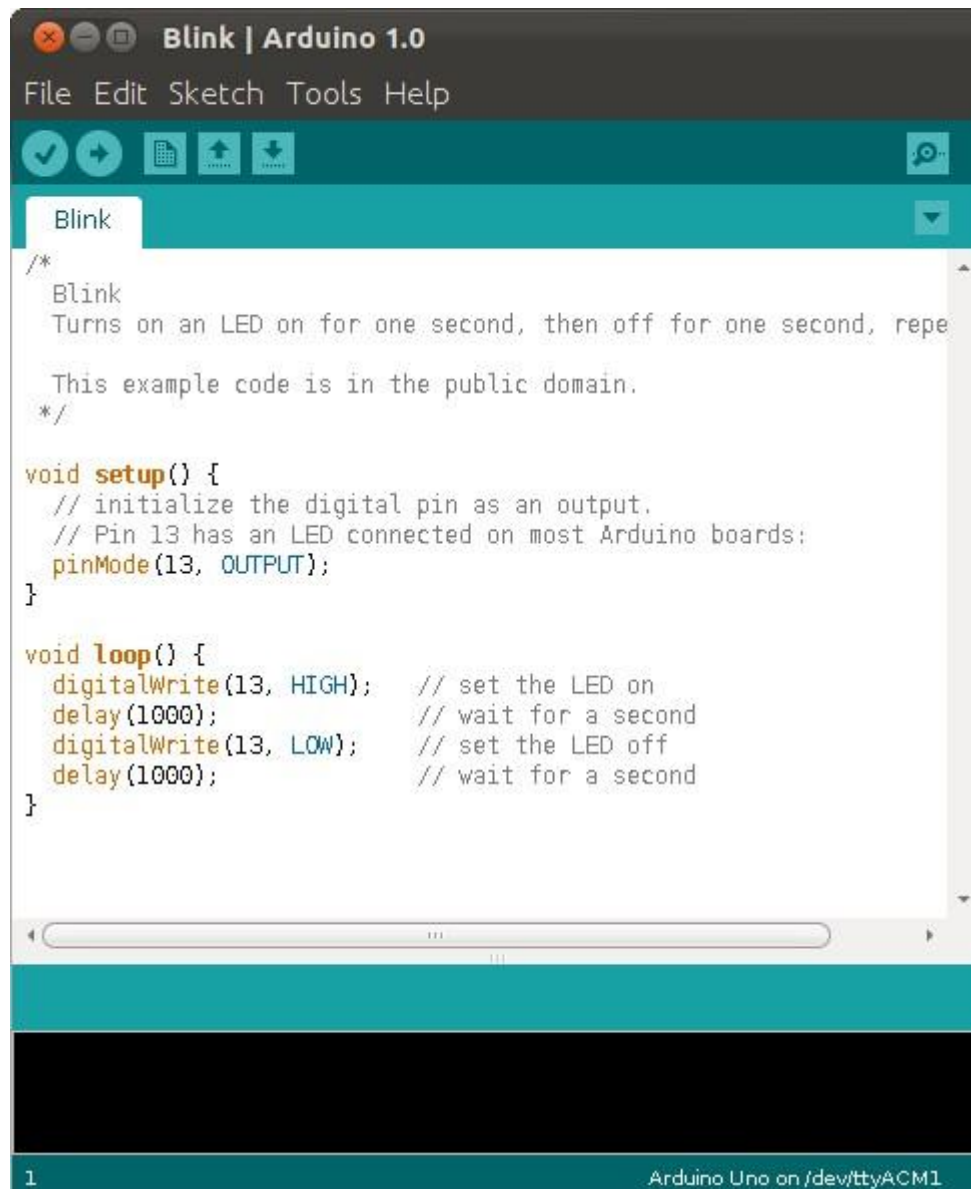


Fig 4.8: Screenshot of Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program:

setup(): a function run once at the start of a program that can initialize settings

loop(): a function called repeatedly until the board powers off unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings.
- loop(): a function called repeatedly until the board powers off.

Hardware requirements:

Hardware and Electronics	Quantity
Arduino Uno Microcontroller	1
Motor(M4,M5)	2
Motor(M2,M3)	2
Motor(M1)	1
SSC-32 Servo controller	1
Power Supply	1
Arm Frame	Kit Bundle

Software requirements:

IDE	Language
Arduino IDE	C, C++

4.1.6 Processing IDE

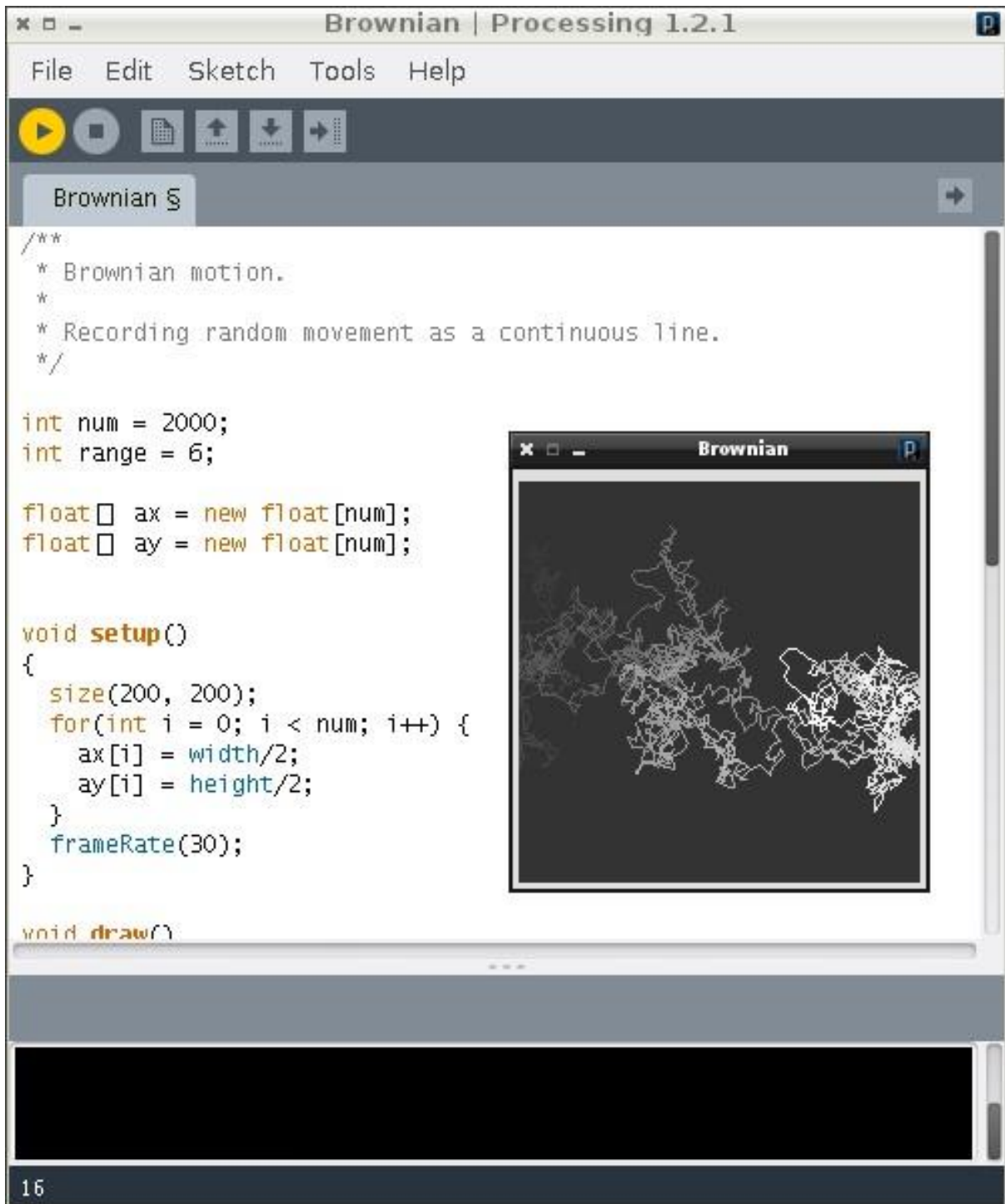


Fig 4.9: Processing IDE

Processing is an open source programming language and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching the fundamentals of computer programming in a visual context, and to serve as the foundation for electronic sketchbooks. The project was initiated in 2001 by Casey

Reas and Benjamin Fry, both formerly of the Aesthetics and Computation Group at the MIT Media Lab. One of the stated aims of Processing is to act as a tool to get nonprogrammers started with programming, through the instant gratification of visual feedback. Processing includes a sketchbook, a minimal alternative to an integrated development environment (IDE) for organizing projects. Every Processing sketch is actually a subclass of the Applet Java class which implements most of the Processing language's features. When programming in Processing, all additional classes defined will be treated as inner classes when the code is translated into pure Java before compiling. This means that the use of static variables and methods in classes is prohibited unless you explicitly tell Processing that you want to code in pure Java mode.

Processing also allows for users to create their own classes within the PApplet sketch. This allows for complex data types that can include any number of arguments and avoids the limitations of solely using standard data types such as: int (integer), char (character), float (real number), and color (RGB, ARGB, hex).

4.2. Implementation Stages

Stage 1 : Making a Layout for Hardware Components and testing them

Stage 2: Generating The code for microcontroller and uploading it on to the device

Stage 3: Generating the Processing code to integrate with the microcontroller

Stage 4: Assembling the hardware components and perform a test as a system

Stage 5: Integrate the controller with the laptop/computer

Stage 6: Execute the code and interface and provide the required commands

Stage 7: Integrate Light sensor with the breadboard

Stage 8: Modification of code so as to implement light sensor

Stage 9: Execute the system with light sensor integrated

CHAPTER-5: Results and Discussions

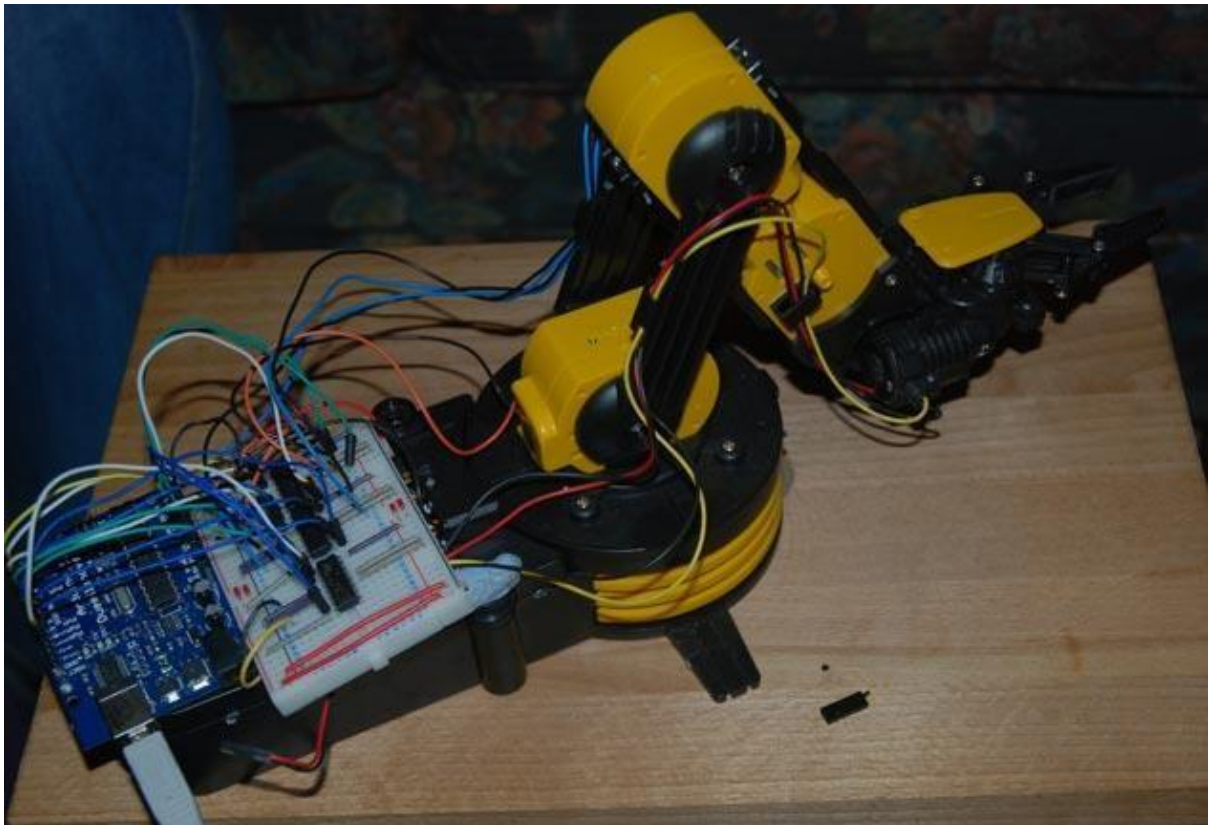


Fig 6.1 : Fully functional Robotic arm

The above photo shows the fully functioning robotic arm. As it is seen, The logical processing for the functioning of the robotic arm is done by the Arduino Micro controller.

This Controller obtains the logic using the code that is fetched. Since the Arduino Uno Microcontroller is connected to the Computer System, it receives

Now, The Interface on the computing system is developed using Processing IDE. As discussed above, it is a special purpose IDE used to design Interfaces. Using the command buttons on the interface, commands can be provided to the Arm.

This arm also implements a light sensor. The logic states that, The arm shall work only if a certain amount of light is detected by the sensor. The sensor is integrated with the breadboard and is further used to provide and input.

Steps to use the arm:

1. Set up the Arm, assemble all the terminals to their respective ports on the breadboard.
2. Set up the microcontroller, assemble the terminals appropriately.
3. Set up the power supply. Use the recommended value for Vcc.
4. Upload The code in the Microcontroller.
5. Run the Processing Code
6. Provide the Light input to the sensor
7. Use the command buttons on the interface to give command to the arm.

Algorithm:

1. Start execution.
2. Scan for input from the interface.
3. If input detected, supply the value to the port, else, go to step 2
4. Pass the value received to the microcontroller.
5. Generate appropriate command signals on the breadboard.
6. Supply the voltage and direction to the DC motors in the Arm and Actuate it
7. Continue till the system is functioning.
8. End.

CHAPTER-6: CONCLUSION

After studying the various concepts of a robotic arm and referring to various research work done on this topic, we have successfully implemented the Microcontroller based robotic arm using the Arduino Microcontroller.

Future scope:

We hope the work done on this project will help provide an easy way of implementing the robotic arm and will create possibilities of creating a much more refined product in the future. Also the project work done should ease the path for implementing Artificial Intelligence in a robotic arm. This will help creating various prototypes which may be helpful in many ways.

APPENDIX A: ABBREVIATIONS

Processing Code

```
import processing.serial.*;
Serial port;
// values to store X, Y for each button
int M1LX, M1RX, M2LX, M2RX, M3LX, M3RX, M4LX, M4RX, M5LX, M5RX; int
    M1LY, M1RY, M2LY, M2RY, M3LY, M3RY, M4LY, M4RY, M5LY, M5RY;

int boxSize = 64;

arrow myRightArrow;
int[]rightArrowxpoints={ 30,54,30,30,0,0,30};
int[]rightArrowypoints={ 0,27,54,40,40,15,15}; arrow
myLeftArrow;
int[]leftArrowxpoints={ 0,24,24,54,54,24,24};
int[]leftArrowypoints={ 27,0,15,15,40,40,54};
PFont myFont; //Font

void setup()
{
    // screen size of the program  size(145,
455);
    // coordinates of each button box
    // base motor M1LX = Motor 1 Left X  etc..
    M1LX = 5;
    M1LY = 25;
```

```

M1RX = 75;
M1RY = 25;
// shoulder motor
M2LX = 5;
M2LY = 115;
M2RX = 75;
M2RY = 115;
// elbow motor
M3LX = 5;
M3LY = 205;
M3RX = 75;
M3RY = 205;
// wrist motor
M4LX = 5;
M4LY = 295;
M4RX = 75;
M4RY = 295;
// hand motor
M5LX = 5;
M5LY = 385;
M5RX = 75;
M5RY = 385;

println(Serial.list()); // set the font
to use myFont =
createFont("verdana", 12);
textFont(myFont);

port = new Serial(this, Serial.list()[0], 9600);

myRightArrow = new arrow(rightArrowxpoints,rightArrowypoints,7); myLeftArrow
= new arrow(leftArrowxpoints,leftArrowypoints,7);
}

```

```

void draw()
{
    background(0);
    noStroke();
    fill(150);

    // draw each box/ button with a label above each
    text("Base Motor (Q/W)", 5, 5, 200, 75);
    text("Shoulder Motor (E/R)", 5, 95, 200, 75);
    text("Elbow Motor (A/S)", 5, 185, 200, 75);
    text("Wrist Motor (D/F)", 5, 275, 200, 75);
    text("Hand Motor (Z/X)", 5, 365, 200, 75);

    if(keyPressed)
    {
        if (key == 'q' || key ==
'Q')
        {
            port.write('Q');
        }
        if (key == 'w' || key ==
'W')
        {
            port.write('W');
        }
        if (key == 'e' || key ==
'E')
        {
            port.write('E');
        }
        if (key == 'r' || key ==
'R')
        {
            port.write('R');
        }
        if (key == 'a' || key ==
'A')
        {
            port.write('A');
        }
    }
}
    
```

```

    }  if (key == 's' || key ==
'S')
    {
port.write('S');
    }  if (key == 'd' || key
== 'D')    {
port.write('D');
    }  if (key == 'f' || key ==
'F')
    {
port.write('F');
    }  if (key == 'z' || key ==
'Z')
    {
port.write('Z');
    }  if (key == 'x' || key ==
'X')
    {
port.write('X');
    }
}

// if no key is pressed check to see if the mouse button is pressed  else
if (mousePressed == true)
{
    // check to see if the mouse is inside each box/ button if so send the value
    if (mouseX > M1LX-boxSize && mouseX < M1LX+boxSize && mouseY >
M1LY-boxSize && mouseY < M1LY+boxSize)
    {
port.write('Q');
    }

    else if(mouseX > M1RX-boxSize && mouseX < M1RX+boxSize && mouseY >
M1RY-boxSize && mouseY < M1RY+boxSize)
    {
port.write('W');

```



```

    }
    else if(mouseX > M2LX-boxSize && mouseX < M2LX+boxSize && mouseY >
M2LY-boxSize && mouseY < M2LY+boxSize)
    {
port.write('E');    }

    else if(mouseX > M2RX-boxSize && mouseX < M2RX+boxSize && mouseY >
M2RY-boxSize && mouseY < M2RY+boxSize)
    {
port.write('R');
    }
    else if(mouseX > M3LX-boxSize && mouseX < M3LX+boxSize && mouseY >
M3LY-boxSize && mouseY < M3LY+boxSize)
    {
port.write('A');
    }
    else if(mouseX > M3RX-boxSize && mouseX < M3RX+boxSize && mouseY >
M3RY-boxSize && mouseY < M3RY+boxSize)
    {
fill(200);
port.write('S');
    }
    else if (mouseX > M4LX-boxSize && mouseX < M4LX+boxSize && mouseY >
M4LY-boxSize && mouseY < M4LY+boxSize)
    {
port.write('D');
    }
    else if(mouseX > M4RX-boxSize && mouseX < M4RX+boxSize && mouseY >
M4RY-boxSize && mouseY < M4RY+boxSize)
    {
port.write('F');
    }
    else if (mouseX > M5LX-boxSize && mouseX < M5LX+boxSize && mouseY >
M5LY-boxSize && mouseY < M5LY+boxSize)

```

```

    {
port.write('Z');
    }
    else if(mouseX > M5RX-boxSize && mouseX < M5RX+boxSize && mouseY >
M5RY-boxSize && mouseY < M5RY+boxSize)
    {
port.write('X');
    } else {
port.write('O');
    } } else
{
port.write('O');
    }

// draw the buttons
myRightArrow.drawArrow(80,30);
myRightArrow.drawArrow(80,120);
myRightArrow.drawArrow(80,210);
myRightArrow.drawArrow(80,300);
myRightArrow.drawArrow(80,390);
myLeftArrow.drawArrow(10,30);
myLeftArrow.drawArrow(10,120);
myLeftArrow.drawArrow(10,210);
myLeftArrow.drawArrow(10,300);
myLeftArrow.drawArrow(10,390);
}

```

Arduino Code

```

int baseMotorEnablePin = 2;
int baseMotorPin1 = 3;
int baseMotorPin2 = 4;          int
shoulderMotorEnablePin = 14;

```

```

int shoulderMotorPin1 = 15;
int shoulderMotorPin2 = 16; int
elbowMotorEnablePin = 8;
int elbowMotorPin1 = 9;          int
elbowMotorPin2 = 10;             int
wristMotorEnablePin = 5;
int wristMotorPin1 = 6;
int wristMotorPin2 = 7; int
handMotorEnablePin = 11;
int handMotorPin1 = 17;          int
handMotorPin2 = 18;

// set a variable to store the byte sent from the serial port int
incomingByte;

void setup() {
    // set the SN754410 pins as outputs:
    pinMode(baseMotorPin1, OUTPUT);
    pinMode(baseMotorPin2, OUTPUT);
    pinMode(baseMotorEnablePin, OUTPUT);
    digitalWrite(baseMotorEnablePin, HIGH);
    pinMode(shoulderMotorPin1, OUTPUT);
    pinMode(shoulderMotorPin2, OUTPUT);
    pinMode(shoulderMotorEnablePin, OUTPUT);
    digitalWrite(shoulderMotorEnablePin, HIGH);
    pinMode(elbowMotorPin1, OUTPUT);
    pinMode(elbowMotorPin2, OUTPUT);
    pinMode(elbowMotorEnablePin, OUTPUT);
    digitalWrite(elbowMotorEnablePin, HIGH);
    pinMode(wristMotorPin1, OUTPUT);
    pinMode(wristMotorPin2, OUTPUT);
    pinMode(wristMotorEnablePin, OUTPUT);
    digitalWrite(wristMotorEnablePin, HIGH);
    pinMode(handMotorPin1, OUTPUT);
    pinMode(handMotorPin2, OUTPUT);
}

```

```

pinMode(handMotorEnablePin, OUTPUT);
digitalWrite(handMotorEnablePin, HIGH);

// start sending data at 9600 baud rate
Serial.begin(9600);
}

void loop() {
    // check that there's something in the serial buffer  if
    (Serial.available() > 0) {
        // read the byte and store it in our variable
        // the byte sent is actually an ascii value
        incomingByte = Serial.read();  // note the
        upper casing of each letter!  // each letter
        turns a motor different way.  if
        (incomingByte == 'Q') {
            digitalWrite(baseMotorPin1, LOW);
            digitalWrite(baseMotorPin2, HIGH);
        }
        if (incomingByte == 'W') {
            digitalWrite(baseMotorPin1, HIGH);
            digitalWrite(baseMotorPin2, LOW);
        }
        if (incomingByte == 'E') {
            digitalWrite(shoulderMotorPin1, LOW);
            digitalWrite(shoulderMotorPin2, HIGH);
        }
        if (incomingByte == 'R') {
            digitalWrite(shoulderMotorPin1, HIGH);
            digitalWrite(shoulderMotorPin2, LOW);
        }
        if (incomingByte == 'A') {
            digitalWrite(elbowMotorPin1, LOW);
            digitalWrite(elbowMotorPin2, HIGH);
        }
    }
}

```

```

    }
    if (incomingByte == 'S') {
digitalWrite(elbowMotorPin1, HIGH);
digitalWrite(elbowMotorPin2, LOW);
    }
    if (incomingByte == 'D') {
digitalWrite(wristMotorPin1, LOW);
digitalWrite(wristMotorPin2, HIGH);
    }
    if (incomingByte == 'F') {
digitalWrite(wristMotorPin1, HIGH);
digitalWrite(wristMotorPin2, LOW);
    }
    if (incomingByte == 'Z') {
digitalWrite(handMotorPin1, LOW);
digitalWrite(handMotorPin2, HIGH);
    }
    if (incomingByte == 'X') {
digitalWrite(handMotorPin1, HIGH);
digitalWrite(handMotorPin2, LOW);
    }
    // if a O is sent make sure the motors are turned off
    if (incomingByte == 'O') {
digitalWrite(baseMotorPin1, LOW);
digitalWrite(baseMotorPin2, LOW);
digitalWrite(shoulderMotorPin1, LOW);
digitalWrite(shoulderMotorPin2, LOW);
digitalWrite(elbowMotorPin1, LOW);
digitalWrite(elbowMotorPin2, LOW);
digitalWrite(wristMotorPin1, LOW);
digitalWrite(wristMotorPin2, LOW);
digitalWrite(handMotorPin1, LOW);
digitalWrite(handMotorPin2, LOW);
    }

```

```
}  
}
```