AUTONOMOUS VEHICLE

By

JAINEEL SHAH

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate Program in Mechanical and Aerospace Engineering

Written under the direction

of

Haim Baruh

and approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2020

ABSTRACT OF THE THESIS

AUTONOMOUS VEHICLE

by Jaineel Shah

Thesis Director:

Professor Haim Baruh

The main goal of this research is to design an autonomous vehicle that can follow any object that it is trained to follow, while giving a live feed of its surroundings to observers. The design of the autonomous vehicle is accomplished by modifying a small scale manually controlled go-kart with motors, electronics and cameras. This thesis concentrates on hardware development for the autonomous vehicle.

The hardware development included design of supporting parts, selection and connection of motors, cameras, wiring the circuit board and all components, adding new features to the car, mounting gears and steering, circuits, and also software development for functioning of the car, coding and adding new features. A platform for major future developments is set with this Autonomous Vehicle, like voice control, live feed, and GPS.

Acknowledgments

I would first like to thank my thesis advisor Professor Haim Baruh of the Mechanical and Aerospace Engineering Department at Rutgers University, New Brunswick. The door to Prof. Baruh's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to thank the experts who were involved in the validation survey for this research project: Professor John Petrovsky and Professor Onur Bilgen. Without their passionate participation and input, the validation survey could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Table of Contents

List of Tables

List of Illustrations

Chapter 1

Introduction

[19]There has been much progress towards driver assistance that requires limited or no driver input. These kinds of systems provide features such as highway lane departure prevention, parking assistance, computer-guided cruise control and vehicle dynamic stability control. The market is not yet ready for a fully autonomous vehicle without driver input. An autonomous vehicle should be tested properly, totally and successfully before it is released to the market.

[20]There are several applications that favor an autonomous vehicle to reduce risk to human life and injury. For example, consider lives of soldiers and human welfare of residents in the conflict zones. Supply caravans that involve the participation of one operator per vehicle are prone to several security problem. Such a caravan greatly increases the risk of casualties for soldiers and support personnel in the area of conflict. However, if the caravan is to be composed of one operator-driven vehicle, either personally or remotely, and followed by autonomous transport vehicles, this could significantly reduce the number of casualties in an attack.

[20]The risk of human injury and life is reduced if the number of support personnel can be reduced. At Rutgers University, to pave the way for the introduction of a

fully autonomous vehicle, a senior design project considered an autonomous vehicle tracking the vehicle ahead of it. This concept can spearhead further technological innovation for a full scale prototype.

[20]In past iterations of the autonomous vehicle senior design projects supervised by Professor Baruh at Rutgers, a radio-controlled car was modified to accept computer-controlled input using data processed with a color-sensing camera. This allowed the autonomous vehicle follow a manually controlled vehicle. In all past iterations of this project, a color sensing camera was used to track a colored object on the lead vehicle. This led to limitations on color choices as certain colors were unable to be used due to lighting and ambient colors of testing environments. Previous iterations of the autonomous vehicle project were similar in that an autonomous vehicle was built from an electric powered remote-controlled car using a single color-sensing camera to support a multi-vehicle caravan. These projects were done using relatively small toy cars and it remained to be seen if these designs could be scaled up to larger platform more in-line with a real world application. An attempt was made in 2015 to increase the size of vehicles used. Instead of implementing a modified remote-controlled car to act as an autonomous vehicle, a small-scale manually controlled go-kart was selected as the vehicle platform for modification. The selected go-cart was the Razor Ground Force Drifter due to its size, higher-power motor and steering characteristics.

In this iteration of the autonomous vehicle, the control circuit has been completely changed from Arduino to a self-created and soldered circuit with various parts like power supply, push button, buzzer, LEDs, LCD screen and microcontroller. The cameras were changed and a better match of cameras for the autonomous car were elected.

There were a lot of options for the microcontroller, like Arduino UNO, Arduino Mega board, ATmega 16, 8051, etc, to be used to carry out these functions. ATmega 16 was the final choice for the microcontroller after careful consideration Arduino and 8051 were the second choices. The entire circuit containing many components like LCD, LEDs, power supply and buzzer, was redesigned, soldered on a board and connected. This was done to increase the simplicity and to design an autonomous vehicle with components that can be obtained easily and at a minimal price compared to an expensive Arduino board that made the circuit and coding more complex.

The camera that was chosen was the Pixy 2 cam cmu5, for its advanced color visualization technology. This camera can learn any object in a matter of seconds and generate an output voltage, LED light output and live feed on a laptop computer. The analog/digital output from two cameras sends a signal to the microcontroller which in turn drives the motors. Two motors were used, one for the forward and backward movement and the other one for steering. HT17-075 8 wire step motors were chosen for their high torque and motor inertia as the chassis these motors had to pull was heavy for other motors that were considered.

The coding was done in BASCOM, a software where the language used is simple and can be used to code complicated functions. The code was such that when both the cameras are trained with the same object, and when both the cameras see the same object, the microcontroller would drive the main motor forward. The camera sends a signal to the microcontroller which drives the steering motor in response. If just the left camera sees the object, the microcontroller drives the steering to the left, and if just the right camera sees the object, the microcontroller drives the steering to right. The car comes to a stop if both cameras don't sense the object they are trained to sense.

The Pixy2 cameras are basically made for Arduino boards and hence the cameras in this circuit had to be powered by a laptop where the feed could be seen live. The cameras are mounted on the front axle so that they move with the steering wheels. Hence, they are in continuous contact with the object they are trained to sense. The LED on the circuit blinks when none of the cameras detect the object and LCD shows the voltage the circuit is receiving from both cameras. The design allows for other sensors and features to be added to the circuit. The system is open to future enhancements.

Testing of the project was first carried out in the laboratory to observe proper working of follower system. The vehicle functioned a bit slow relative to an actual real-sized car, so the design here can be considered as a prototype for future models. Once all the functions of the vehicle were checked properly, the vehicle was ready to be tested on the road following a human controlled vehicle by training the cameras to sense a vehicle.

There were a few problems that were faced. The Pixy2 cam cmu5 comes in two versions and the two cameras that were used in the beginning were not of the same version. Because of this, the microcontroller sensed only one of the cameras that gave analog/digital output. The forward motion was carried out by the Go-Kart's DC Motor, MY1016, 24VDC 36V, which was connected to a relay, which connected the motor to the circuit to receive signal to move forward. The wires used for battery connection were unable to hold the amount of voltage and load running through the wires and they burned. They were replaced with thicker wires with a switch connection that was soldered. In the beginning, the motors were unable to propel the vehicle and were replaced with new motors with a greater torque and power replaced the old ones. As the cameras were Arduino based, their connection to this non-Arduino circuit had to be established in a way they could send a signal to the microcontroller. The cameras were first placed on a mount that was earlier in 2015 but that didn't help as the cameras needed to move with the car's right and left motion. Therefore it was decided to place them near the wheel axle. The coding was rewritten as one of the errors caused a loop due to which the motor kept on running without any sort of signal from the cameras.

The future of this vehicle is very broad can be used as new features can be added and more refined control laws. Some of the features that can be easily added are voice control, proper braking and speed control, a GPS so that the vehicle follows a known path, and additional sensors for new functions. In the long run, as this system develops, space will become limited on the go-kart

platform and this will require the project to move towards a larger vehicle. A next step can either be the implementation of a competition go-kart, or a mini-indy style race car, such as the cars developed by the Rutgers Formula Racing Team.

One of the major future uses of this car would be in the operation of supply transport caravans. Usually caravans require a minimum of one vehicle operator per vehicle in order to function. This does not include any protection detail or support personnel per vehicle in order to function. Such action greatly increases the casualty risk for soldiers and support personnel in a conflict area. However, if the caravan were to be composed of one operator-driven vehicle either personally or remotely or by voice control, and a caravan of autonomous transport vehicles following the lead vehicle, this can reduce the risk of casualty.

Chapter 2

Parts of the Autonomous Vehicle

2.1 Razor Force Ground Drifter Go-Kart

The Razor Force Ground Drifter was purchased from a commercial vendor for $270. It is an electric-powered kart. As the name suggests, it is a kart with ground force design which gives it more stability, balance, and better tires to support the vehicle motion. It has a variable-speed motor that is chain driven. The vehicle can attain speeds of up to 12 miles per hour. The chassis is race-tuned and the rear wheels are super-smooth. The kart runs for 40 minutes per charge and can carry up to 140 pounds of cargo, which to add more sensors, bigger motors, cameras and steering. Everything was removed from the go-kart, the chassis, the main motor and the tires. The chassis was the platform for the circuit, steering, power supply, motors and batteries to be installed. The camera rack and the camera rack bracket were installed in the front part of the chassis.

The main motor of the go-kart was powered by two 12 V batteries. This motor was connected to the main circuit, and included in the control code. The motor shaft is connected to a chain which connects and drives the rear wheels. Please refer to APPENDIX 1 for more information about the Razor Force Ground Drifter.

2.2 Pixy 2 CMU cam5

[2]The Pixy 2 CMU cam5 is smaller, faster and more capable than the original Pixy. Just like Pixy, it can learn to detect objects that one trains it, by just pressing a button. In addition, Pixy2 has new algorithms that detect and track lines, as well as intersections and “road signs”. The road signs can tell the robot what to do, such as turn right, left, slow down, etc. The Pixy 2 performs all these functions at 60 frames-per-second. *Figure 2.1* shows the Pixy CAM Cmu5.

An USB cable connects the camera with a laptop, making it possible to see the live feed, make changes to the configuration, interface, and train it to learn an object. The Pixy2 has several interfaces including SPI, I2C, UART with simple communications, so one can quickly get their controller talking to the Pixy2. The Pixy2 uses a color-based filtering algorithm to detect objects. Color-based filtering methods are fast, efficient and relatively robust. The camera calculates hue and saturation of each RGB pixel from the image sensor and uses these as the primary filtering parameters. Except for the changes in lighting and exposure, the hue basically remains the same. These changes in lighting and exposure can cause problems and effects on the algorithms and cause them to not work. Pixy 2’s filtering algorithm is robust in the matter of lighting and exposure changes.

The camera is trained using the following steps:

1) Press the button on the top until the LED on the camera turns red, and release the button. The LED then starts displaying different colors while blinking.
2) Once a color on the LED resembles the object, press the button on the camera again. The image seen on the laptop at this time is in pixels as the camera is learning the object. The smoother, closer and well-defined object the better is the object recognition by the camera.
3) Once satisfied with image of the object on the laptop, press the button on the camera once and the camera has learned the object.

The interface used for this particular project was analog/digital I/O. This interface gives a voltage (a signal,1) up to 2V from the PIN 1 of the camera when it detects the object and gives 0 volt output (0), when that particular object is not detected. Whether the camera is detecting the object or not is ascertained from the LED. The LED stays on with the object's color when it detects the object and turns off when it doesn't detect the object. PIN 1 of the left camera is connected to the PORTA0 (PIN 40), PIN 1 of the right camera is connected to the PORTA1 (PIN 39), PIN 6 of both cameras are ground pins and are connected to the circuit common ground. Both the cameras are connected to the laptop for power, as the cameras are Arduino-based. Hence they need power supply from the laptop plus providing a live feed from the camera on the laptop screen, which could even be projected at any laptop (by the use of team viewer) while the vehicle is moving.

The interface of the camera is set on the Pixymon V2 software which controls the camera. All the live feed from the camera is viewed on the Pixymon V2 software.

The camera is trained on this software on the laptop. *Figure 2.2* shows the circuit diagram camera for connections. The camera can be trained to follow many objects, colors and signatures, and this training can be saved in the Pixymon V2 software and can be loaded on the camera whenever needed. *Figure 2.3* shows Pixymon V2 software screenshot.

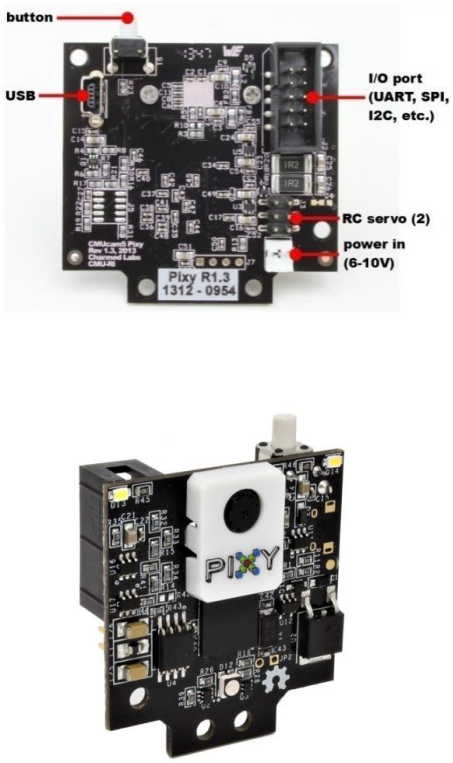


Figure 2.1. Pixy CAM Cmu5 Photo. These Two Images Were Taken From The Manufacturer's Website, www.pixycam.com.

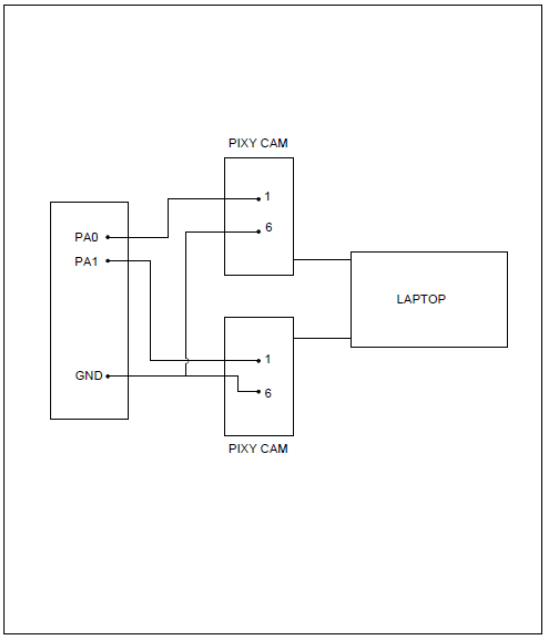


Figure 2.2. Camera Connections Circuit Diagram. *This Diagram Shows All The Wire Connections Of Both The Cameras With Laptop And The Main Circuit.*

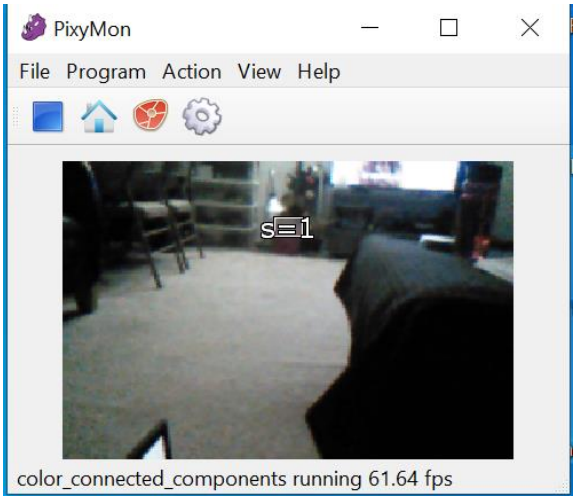


Figure 2.3. Pixymon V2 Software Screenshot. *This Is The Software Where The Camera Projects The Live Feed Of The Surroundings.*

2.3 Power Supply

Every power supply must obtain the energy it supplies to its load, as well as any energy it consumes while performing that task, from an energy source.

Depending on its design, a power supply may obtain energy from:

- Electrical energy transmission systems. Common examples of this include power supplies that convert AC line voltage to DC voltage.
- Energy storage devices, such as batteries and fuel cells.
- Electromechanical systems, such as generators and alternators.
- Solar power.

Commonly specified power supply attributes include:

- The amount of voltage and current the power supply can supply to its load.
- The stability of output voltage or current under varying line and load conditions.
- The length of time the power supply can supply energy without refueling or recharging (applies to power supplies that employ portable energy sources).

Figure 2.4 shows the circuit diagram of the power supply circuit connections.

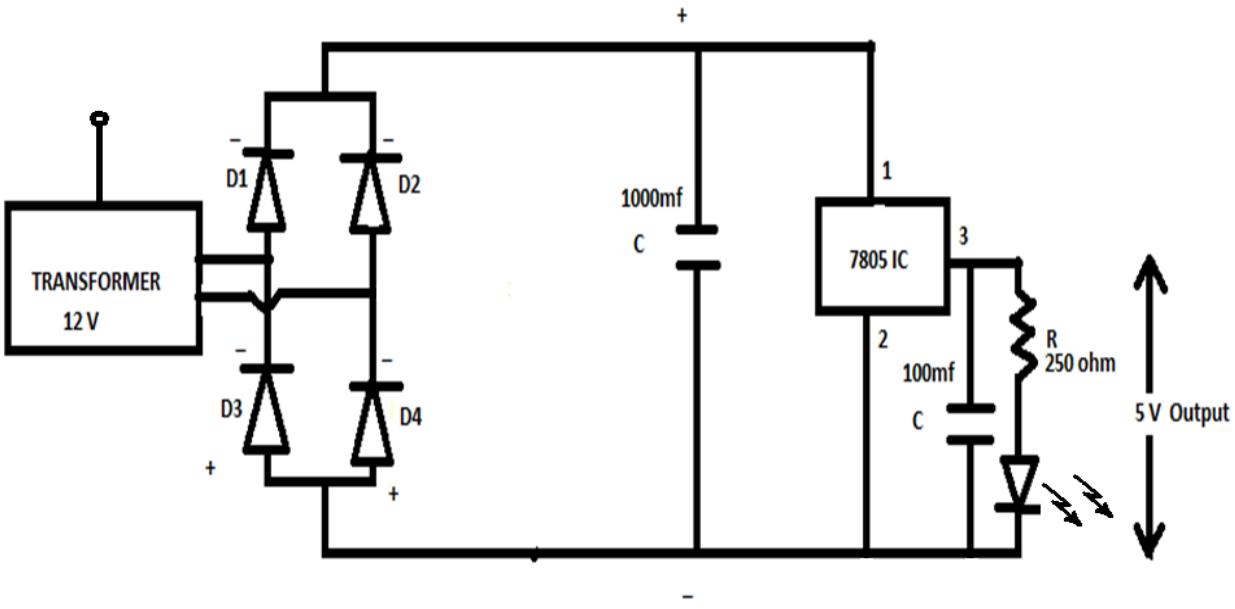


Figure 2.4. Circuit Diagram Of The Power Supply Used In The Autonomous Car.

2.4 ATMEGA 16

A microcontroller is a chip which has a computer processor with all its support function (clock and reset), memory (both program and data), and I/O (including bus interface) built into the device. These built-in functions minimize the need for external circuits and devices in the final application.[4]

The ATMEGA16 Microcontroller gets connected to the power supply and all the motors and every part of the circuit as the ATMEGA16 controls all of it, on the main circuit board. *Figure 2.5* shows a photo of the ATMEGA 16 microcontroller used for the autonomous vehicle.

ATMEGA 16 MICROCONTROLLER:

[4]As indicated in the manufactures brochure the ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

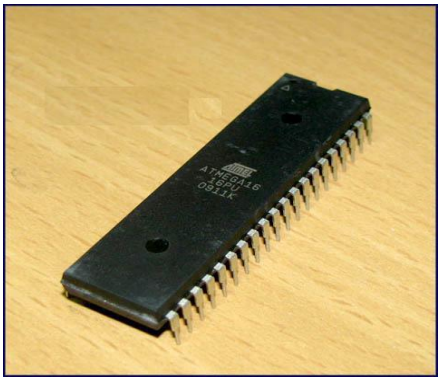


Figure 2.5. Photo of ATMEGA 16 microcontroller used for the autonomous car.

The AVR core combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code-efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits. The boot program can use any interface to download the application program in the Application Flash memory.

Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.Please refer to *APPENDIX 2* for details on features of ATMEGA 16 microcontroller.

2.5 PIN Descriptions

The PIN diagram is shown in *Figure 2.6*. The ATMEGA 16 PIN consists of the following parts as documents by the manufacture.

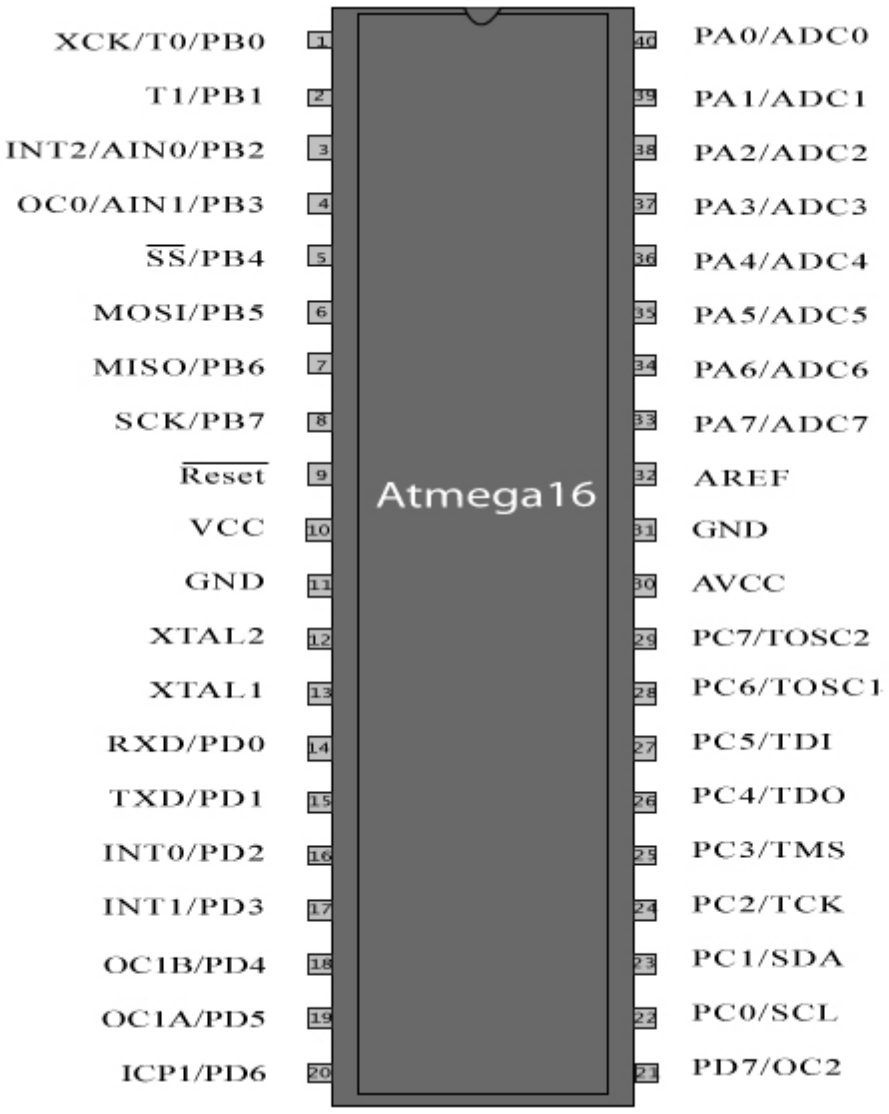


Figure 2.6. Pin Diagram. This diagram was taken from Google, PIN Diagram for ATmega16. Please refer to *APPENDIX 3* for details on PIN Descriptions and BASCOM Software which was used for coding of these PINS.

2.6 ROBOKITS AVR Programmer

A STK-500 loader was used to load the hex file of the program in the microcontroller ATmega16. This was done by means of ROBOKITS AVR programmer. This approach can also be used for various other microcontrollers. *Figure 2.7* shows an image of Robokits Loader that was used to load the code on the microcontroller of the autonomous vehicle. *Figure 2.8* shows a screenshot of the ROBOKITS software used to load the code from the laptop to the microcontroller.

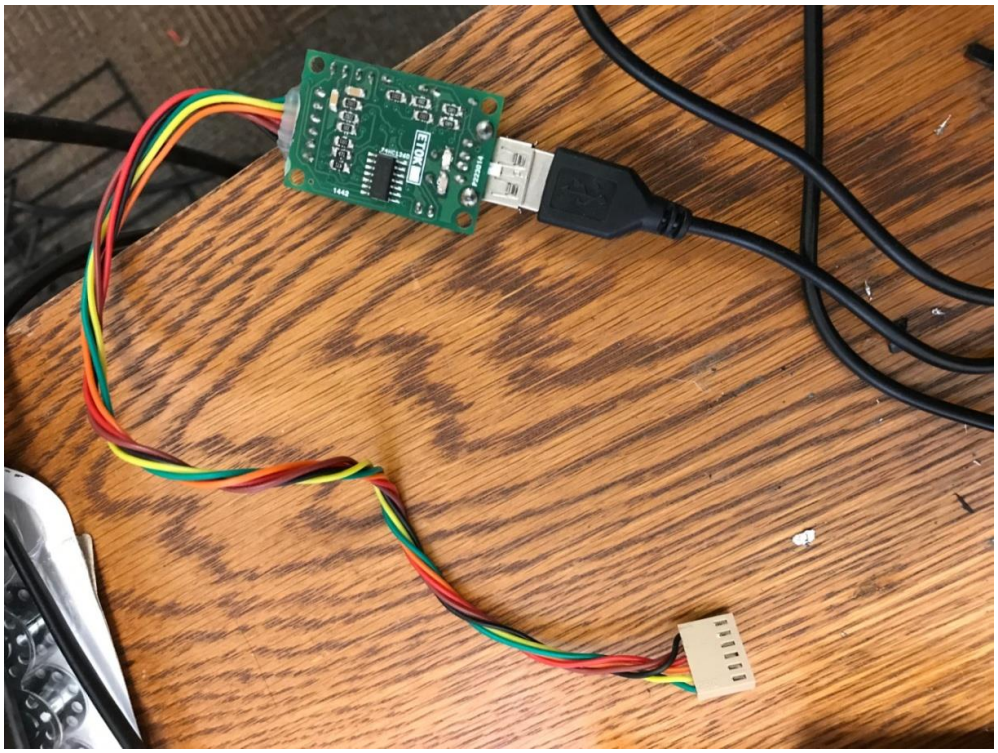


Figure 2.7. Robokits Loader-The Device Used To Load The Code In The Microcontroller.

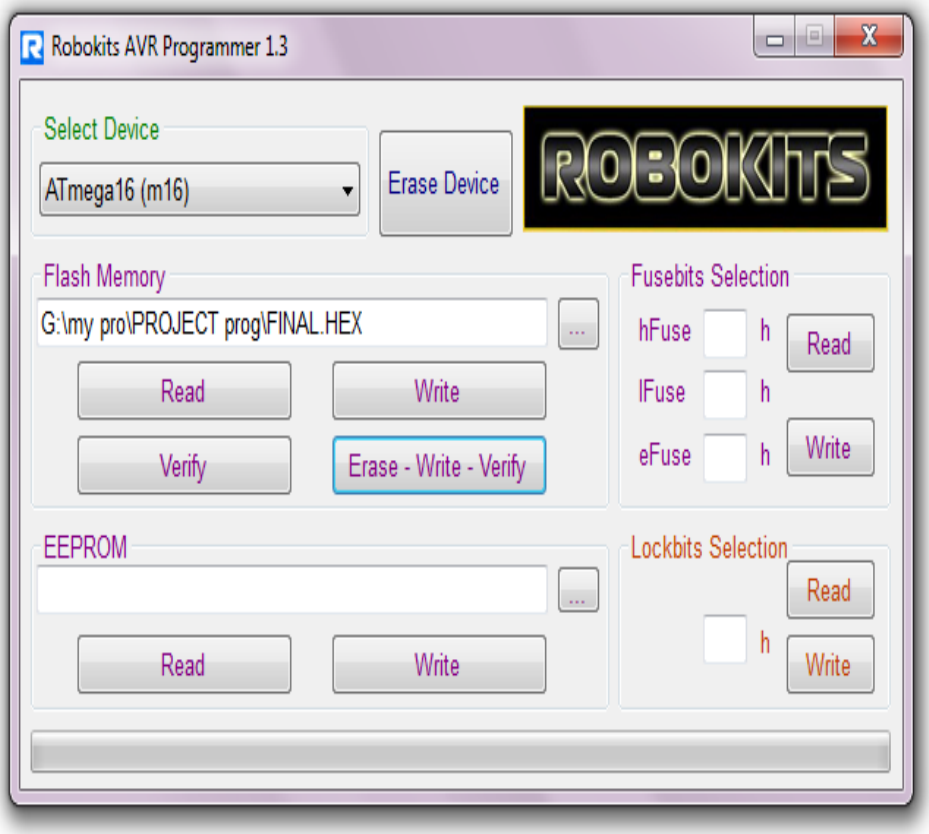


Figure 2.8. Robokits AVR Loader Software Screenshot.

2.7 Stepper Motor

The stepper motor chosen for this particular car was the HT17-075 8 wire stepper motor. This stepper motor was chosen for its high torque and motor inertia as the chassis these motors has to pull was heavier than other motors. [6]The steering stepper motor was installed with a small gear on the motor shaft fixed to it to control the steering motion. For the stepper motor connection, unipolar series connection was used in which the four main driving wires were a identified by checking the voltages and coil connections in the motor.

The other four wires were soldered to the same ground connection as the circuit. To drive the motors, a perfect sequence had to be found to correlate outputs the appropriate step of the motor. This completes the cycle and drives the motor. The code was written so that it identified the sequence of the steps. The stepper motor drivers were used for each motor to give the output.

The HT17-075D two-phase stepper motor w/Double Shaft is suitable for a wide range of motion control applications. Terminated with 8 motor leads, the motor can be connected in different ways, including bipolar series and bipolar parallel. [10]Features of HT17-075D stepper motor include :

- 2-phase hybrid step motor
- High torque design
- Standard NEMA 17 dimensions
- Series and Parallel wiring

Please refer to *APPENDIX 4* for stepper motor selection and connections.

2.8 L298N Motor Driver

The four wires of the stepper motor are connected to the four outputs of the motor driver. The 5V, 12V and ground pins of the driver are connected to the 5V, 12V and ground pins of the circuit. The connections are shown in *Figure 2.9.* *Figure 2.10* shows the photo of the L298N motor driver used for the Autonomous Vehicle. Please refer to *APPENDIX 5* for L298N Features and Specifications.

Circuit Diagram:

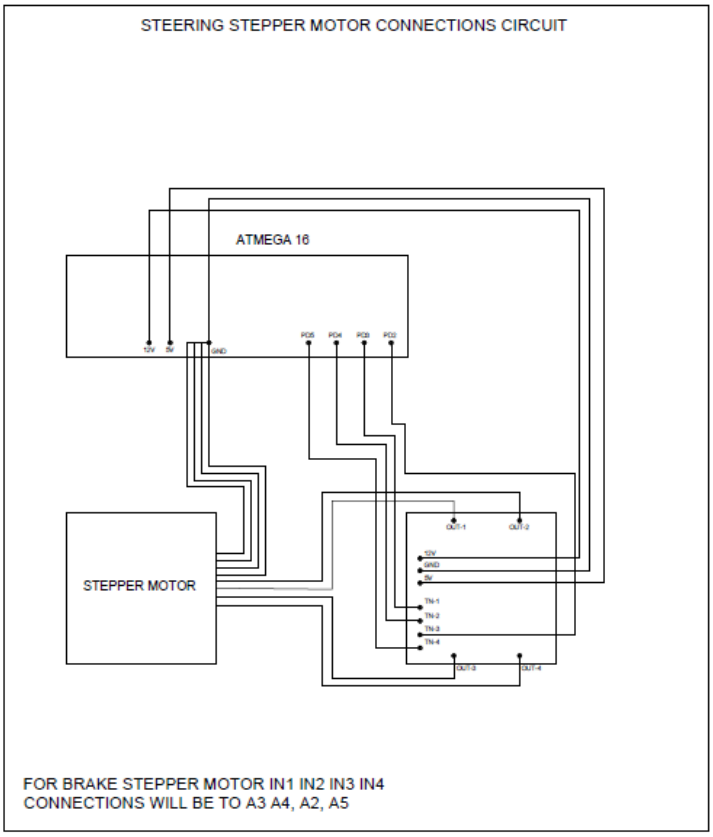


Figure 2.9. Stepper Motor And L298N Motor Driver Connections Circuit.

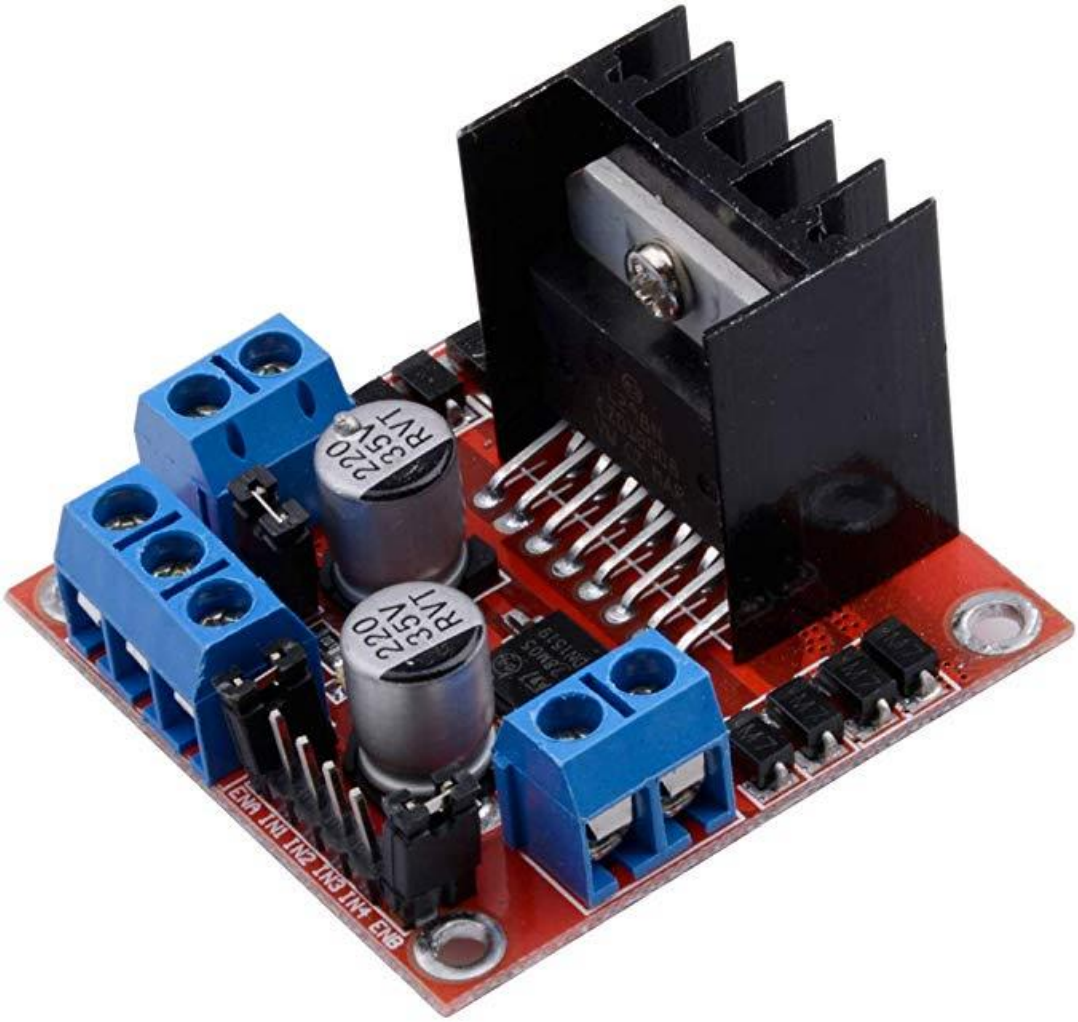


Figure 2.10. L298N Motor Driver Used On The Autonomous Vehicle.

2.9 Relay Module and Go-Kart Motor

Relay Module

[3]A relay is an electromagnetic switch that is used to turn on and turn off a circuit by a low power signal. It is also used when several circuits must be controlled by one signal.

[5]The two common pins of a relay are connected to the two wires of the main DC motor as shown in *Figure 2.11*. The two NO pins are connected to each other which is connected to the positive of the battery and the two NC pins are connected to each other which is connected to the negative of the battery. The VCC pin connects to 5V in the circuit, the ground pin connects to the ground in the circuit and the two other pins connect to the two pins in the circuit controlling the forward and backward motion of the motor. Please refer to *APPENDIX 6* for Relay Module Specifications.

Relay Connection:

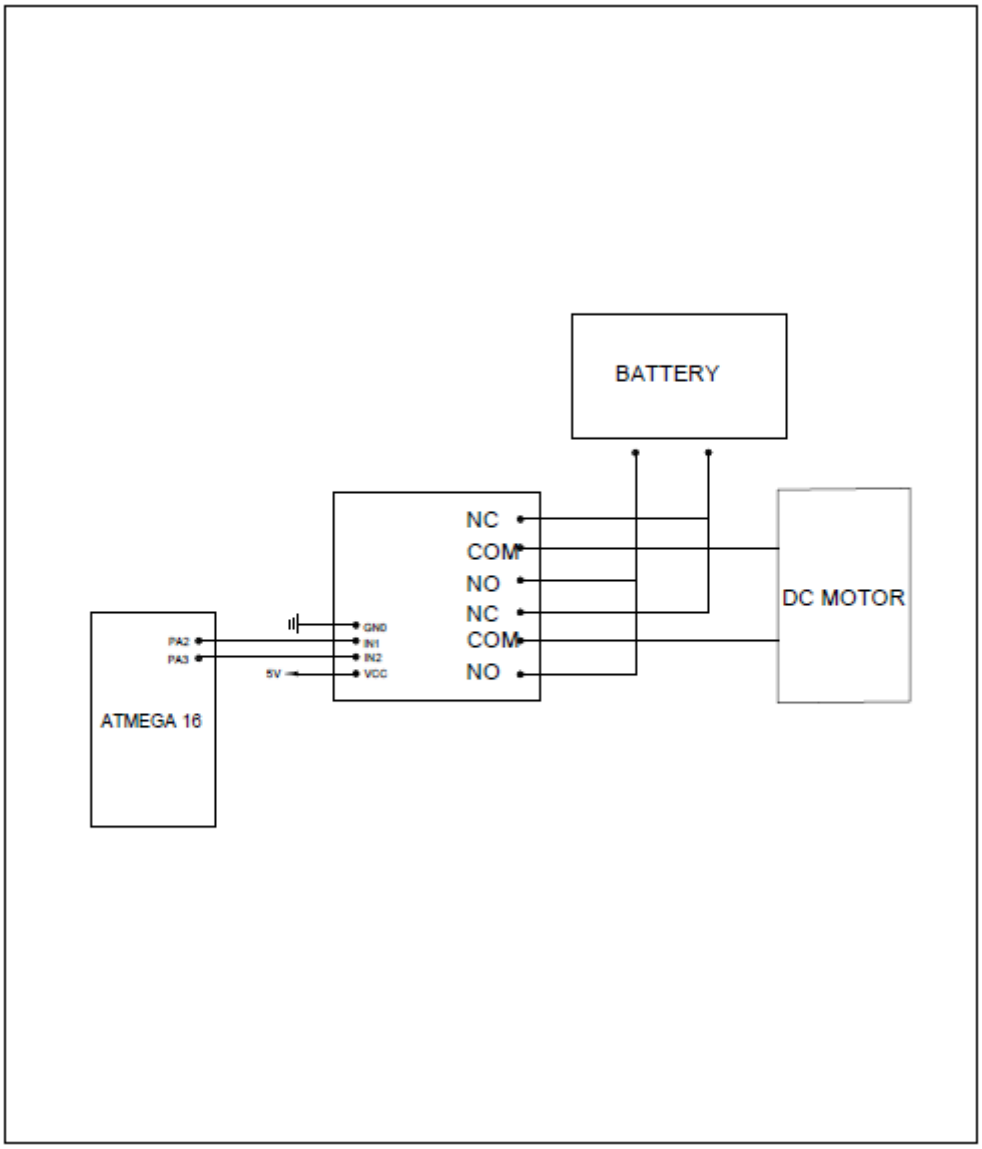


Figure 2.11. Relay Module Connection Circuit Diagram.

2.10 Design of all the Custom made parts

This section describes designs of all the custom parts that were manufactured for support of the stepper motors and the cameras. The figure shown are Solidworks 3-D model, 2-D drawings and photos of the manufactured parts assemblies.

Figure 2.12 shows Solidworks model of Steering Motor Mount Base.

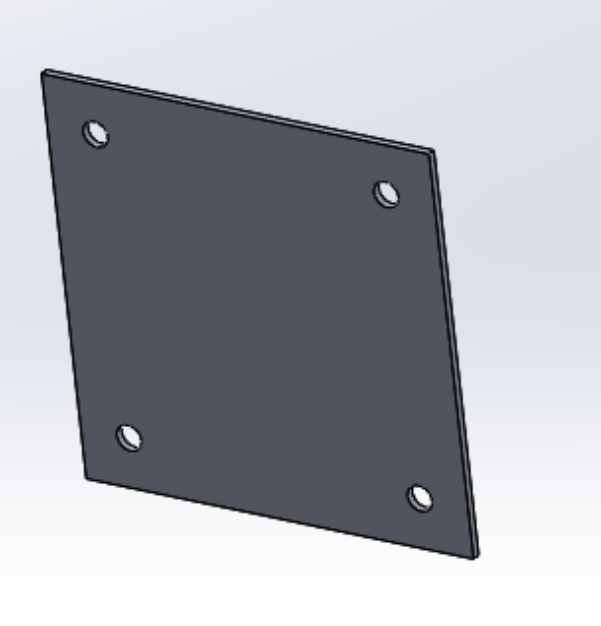


Figure 2.12. Steering Motor Mount Base Solidworks Model.

Figure 2.13 shows Solidworks drawing of Steering Motor Mount Base.

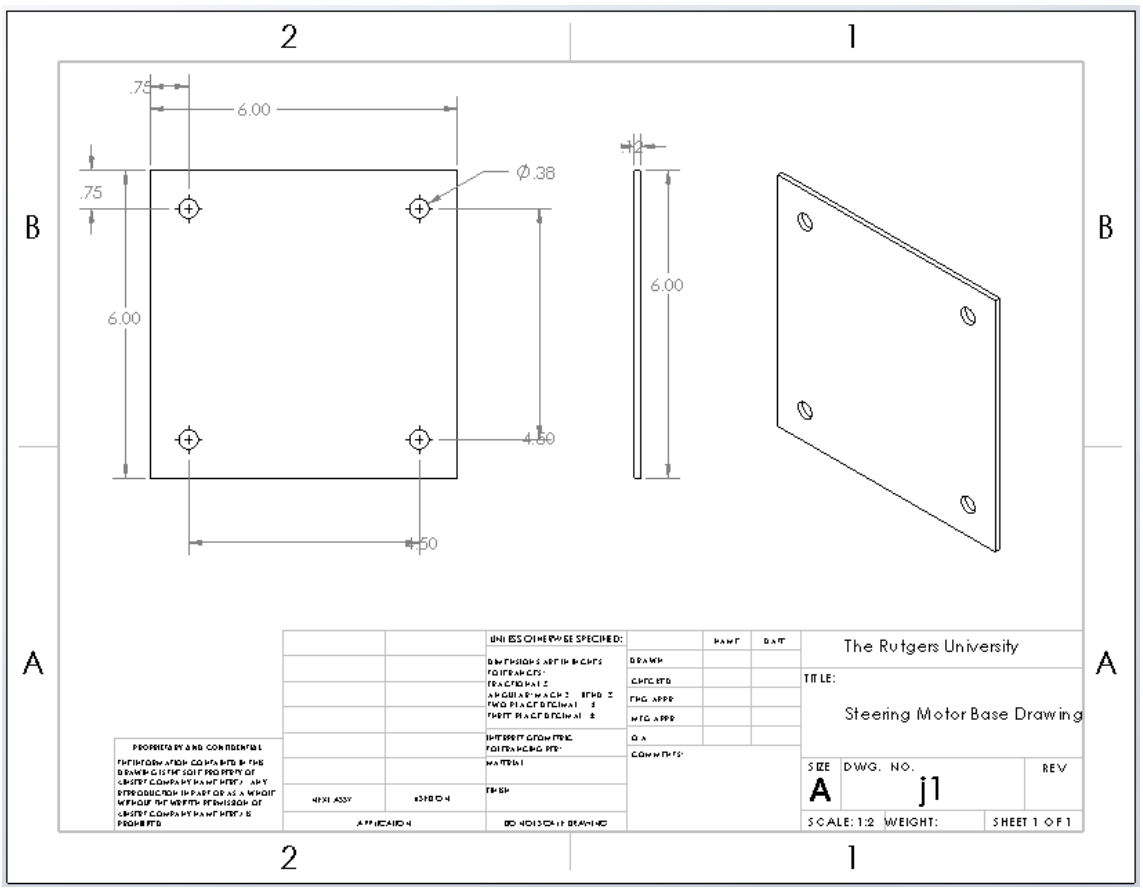


Figure 2.13. Drawing For Steering Motor Mount – Base.

Figure 2.14 shows Solidworks model of Steering Motor Mount Tower.

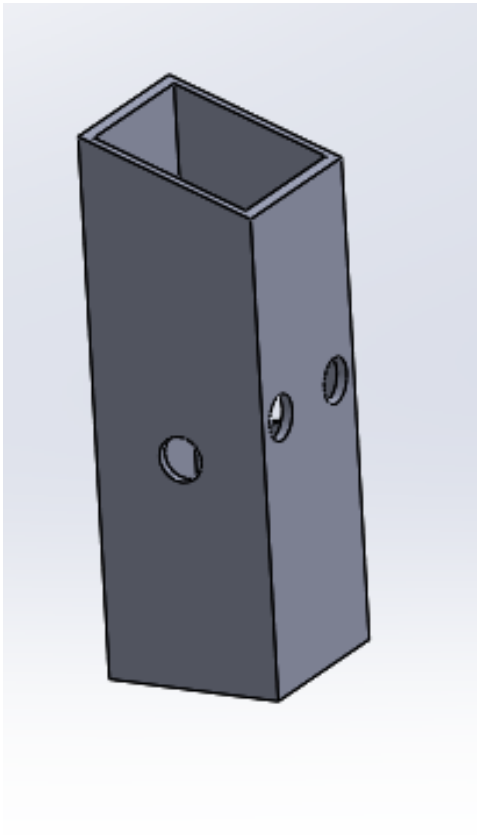


Figure 2.14. Steering Motor Mount – Tower.

Figure 2.15 shows Solidworks drawing of Steering Motor Mount Tower.

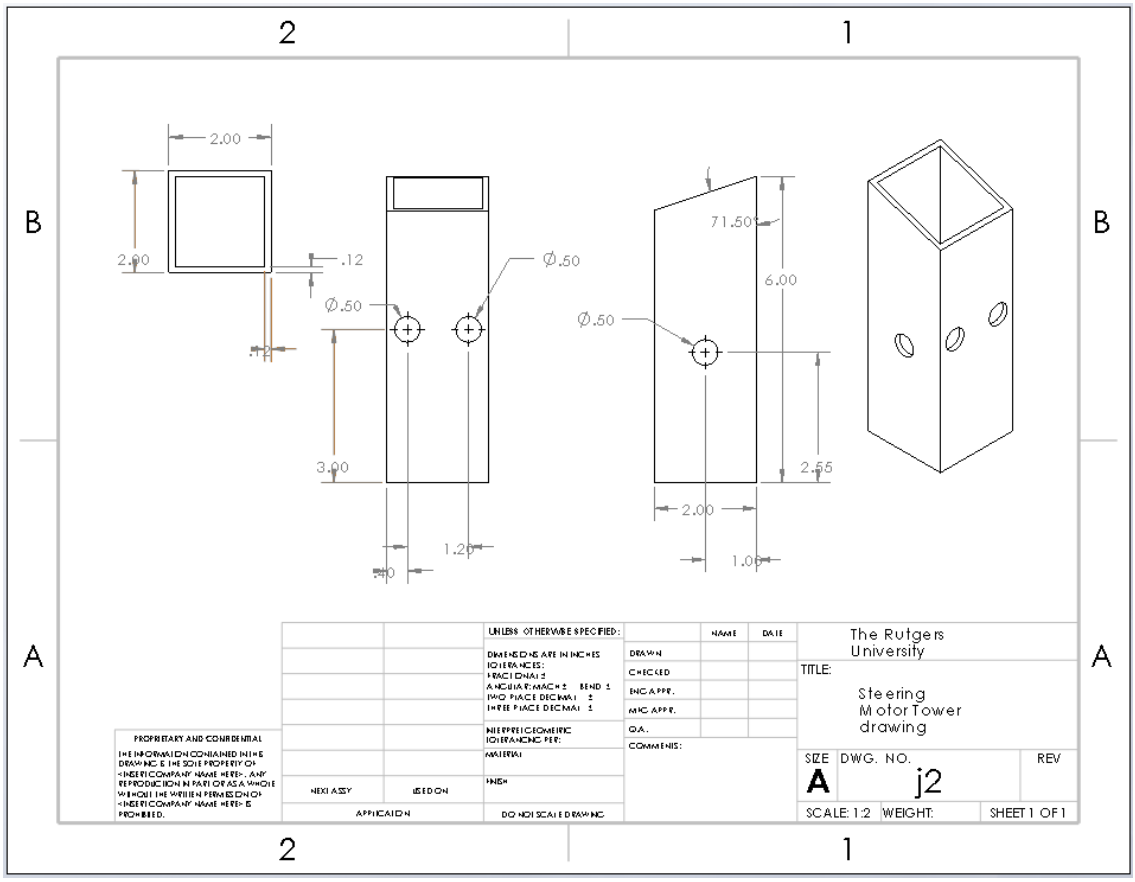


Figure 2.15. Drawing For Steering Motor Mount – Tower.

Figure 2.16 shows Solidworks Model of Steering Motor Mount Clip.

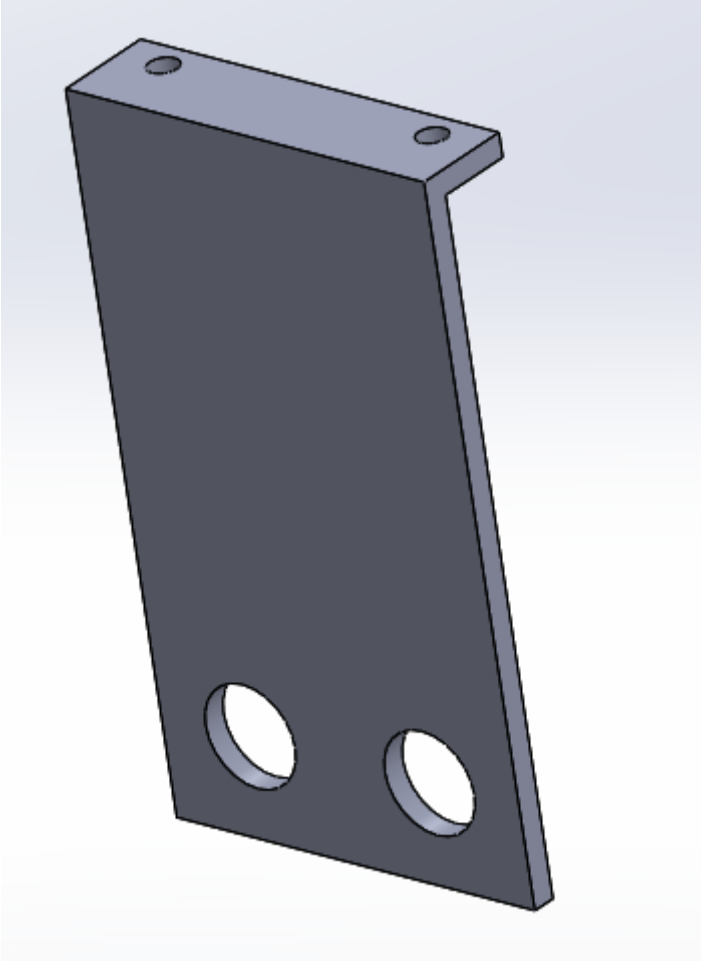


Figure 2.16. Steering Motor Mount - Clip

Figure 2.17 shows Solidworks drawing of Steering Motor Mount Clip.

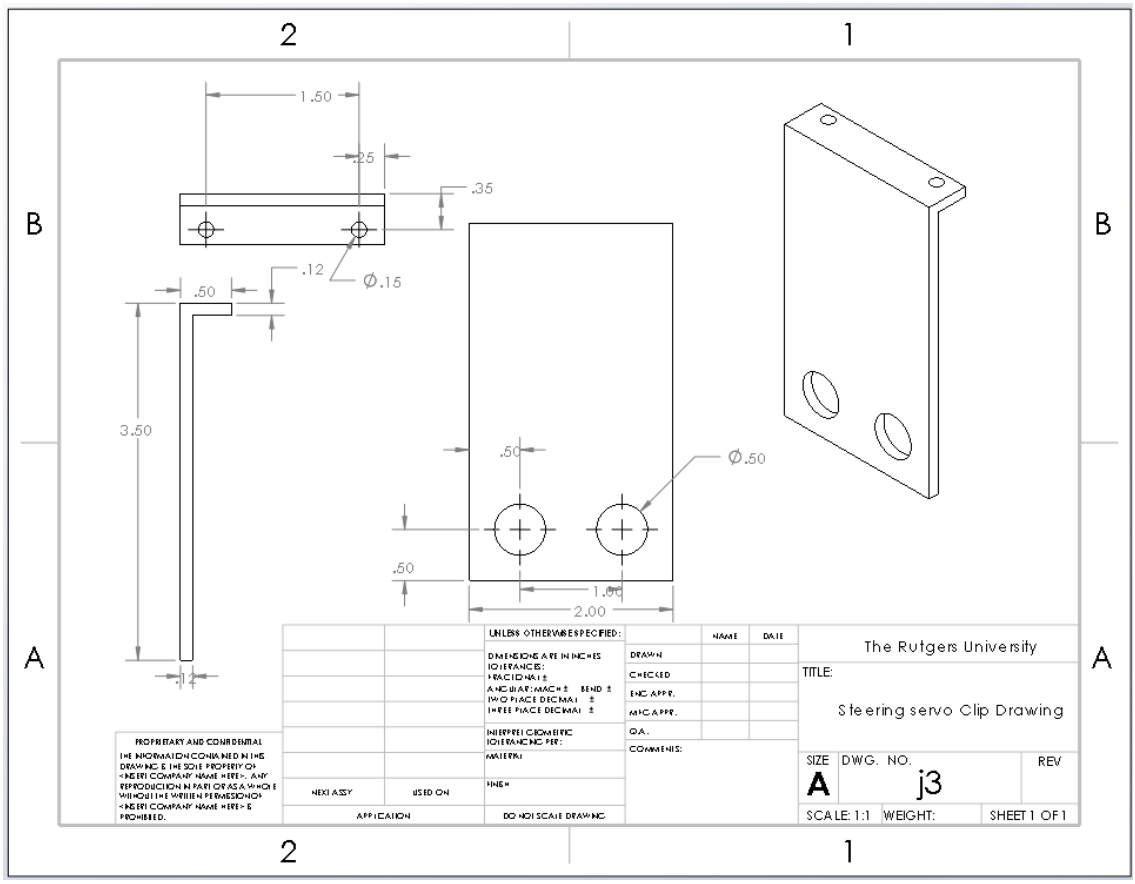


Figure 2.17. Drawing For Steering Motor Mount – Clip (X 2)

Figure 2.18 shows Solidworks model of Camera Mount Rack.

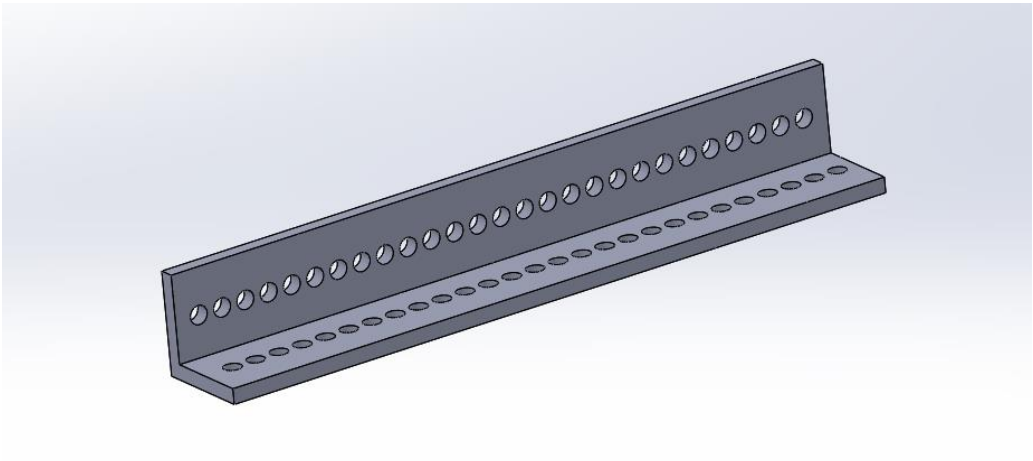


Figure 2.18. Camera Mount Rack

Figure 2.19 shows Solidworks model of Camera Rack L – Bracket Mounts.

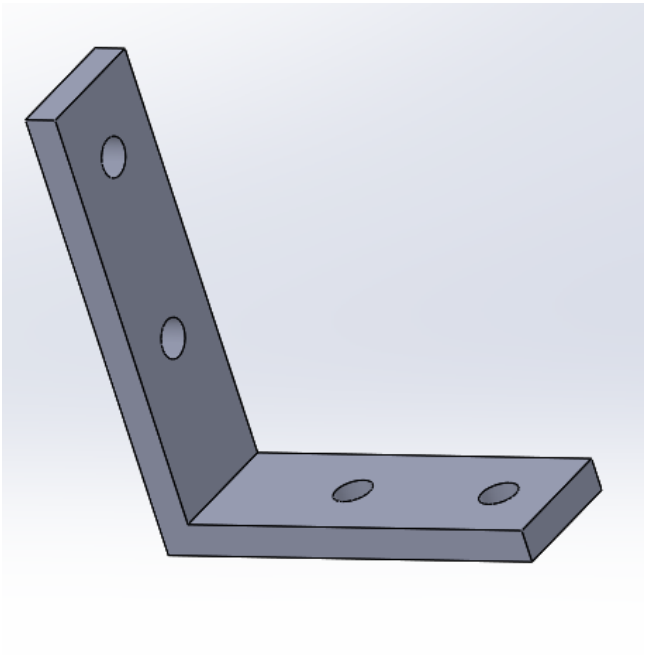


Figure 2.19. Camera Rack L – Bracket Mounts (x 2)

Figure 2.20 shows Solidworks model of Steering Motor Mount Assembly.

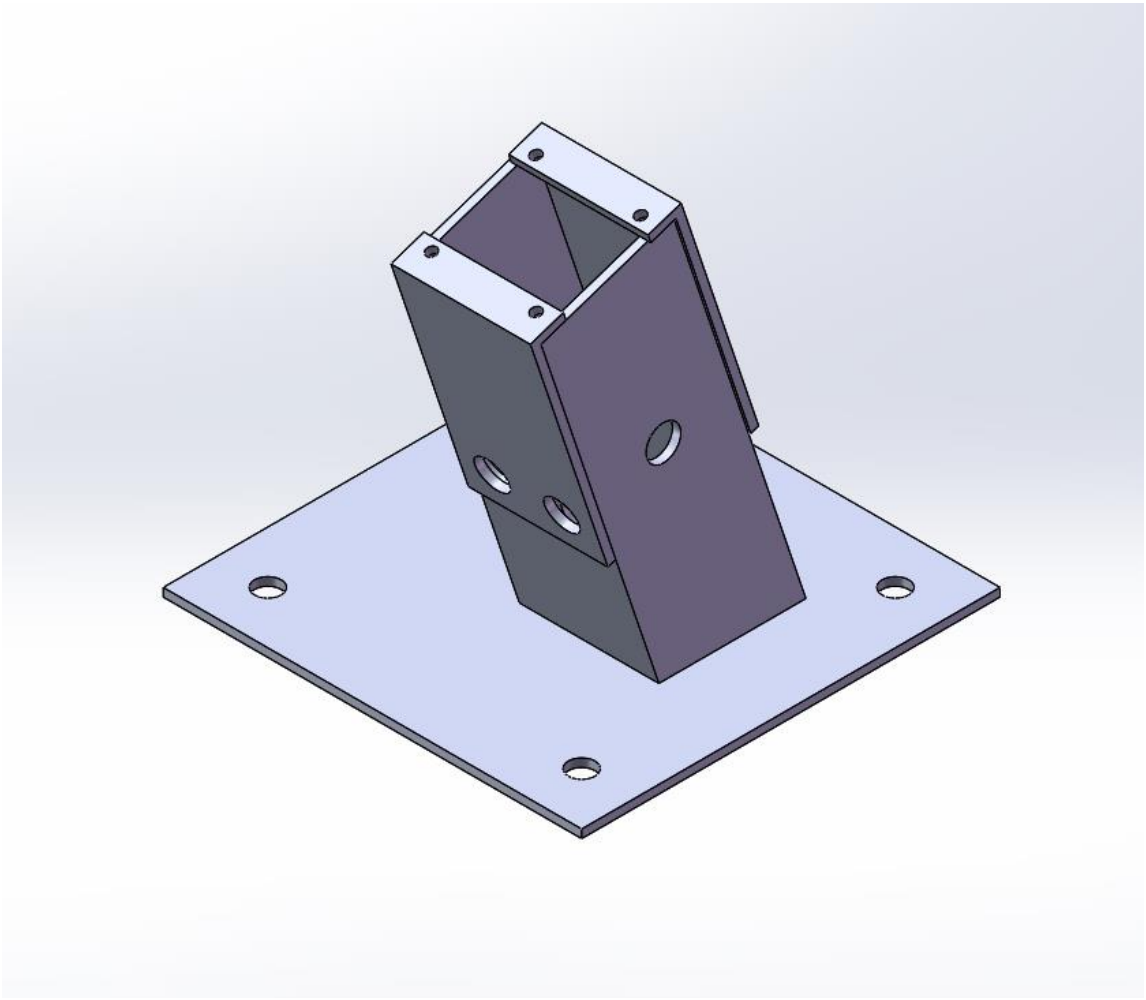


Figure 2.20. Steering Motor Mount Assembly

Figure 2.21 shows Top View of Steering Motor Mount Assembly and circuit connections.

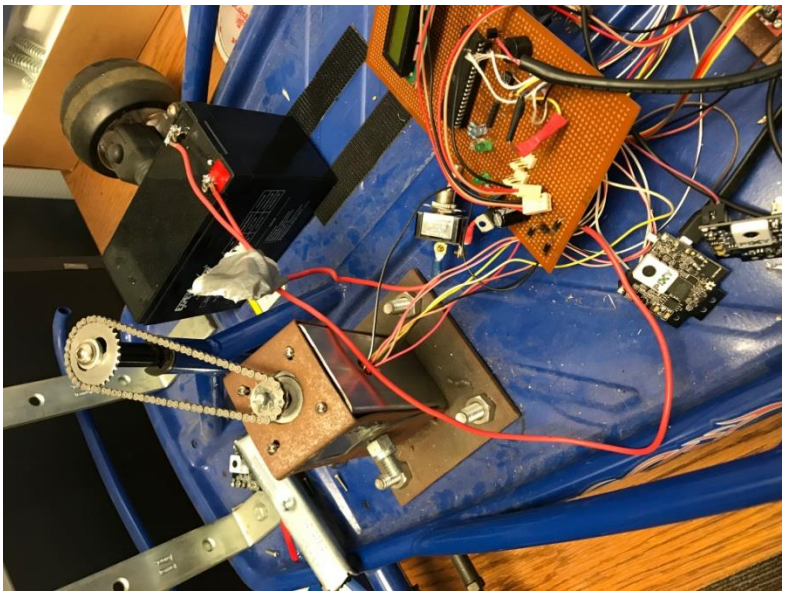


Figure 2.21. Top View Of Steering Motor Mount Assembly And Circuit Connections.

Figure 2.22 shows photo of Autonomous Car.

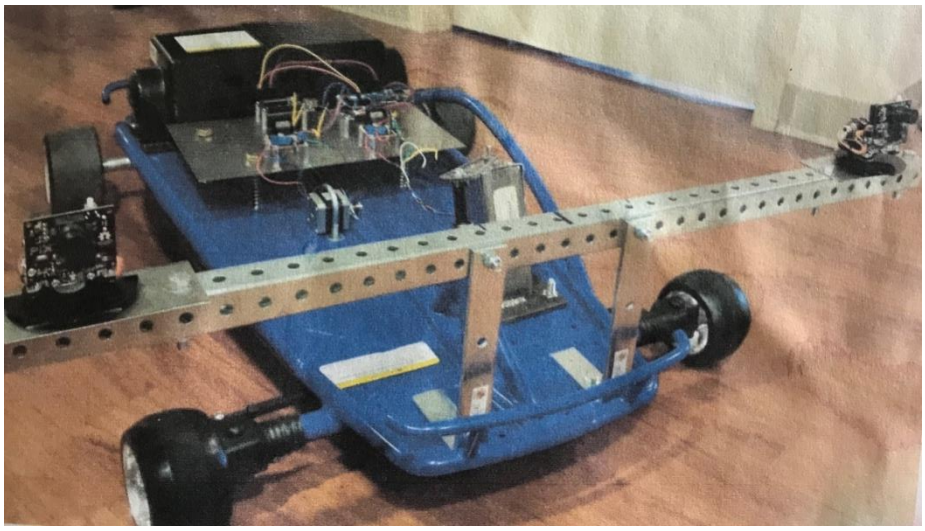


Figure 2.22. Autonomous Car Photo.

2.11 List of Parts Used

Table 1 lists the parts that were purchased to construct the follower vehicle.

Item	Supplier	Catalog #	Quantity	Unit Price	Total
Razor Ground Force Drifter	Walmart	70083524	1	$320.00	$320.00
Pixy 2 (CMUcam5)	Charmed Labs	N/A	2	$60.00	$120.00
Stepper Motor	Ebay.com	N/A	2	$25.00	$50.00
Camera Rack	McMaster-Carr	KD-ERW	1	$44.24	$44.24
Camera Rack Bracket	McMaster-Carr	1556A39	2	$3.05	$6.10
Stepper Motor Controllers	Amazon.com	B00CAG6GX2	2	$8.49	$16.98
Razor Motor Batteries	Amazon.com	B00DL5I8GO	2	$46.99	$93.98
Lead Vehicle	Amazon.com	B00HVBVNVG	1	$139.75	$139.75
Loader	Amazon.com	N/A	1		
Soldering Kit	Amazon.com	N/A	1	$20	$20
Relay Module	Amazon.com	N/A	1	$10	$10

Total Cost = $695.05

Chapter 3

Design Procedure

3.1 Building the Vehicle

In previous design projects iterations were done with an Arduino-based circuit and coding with imaging of a color sensed object by a single camera. Listed below are the many problems in previous versions of the autonomous car that were redesigned and recoded.

- The soldering of all wires and connections were done very weakly that could cause a short circuit.
- The code written was not working to drive the vehicle.
- The cameras had limitations on the functioning of the autonomous car.
- The cameras were not trained and connected for functioning.
- The motors were not working nor did the batteries.
- The main car motor was not connected and not working.

The entire Arduino circuit and the old cameras were removed and a new circuit was designed and coded from the scratch with a completely different non-Arduino circuit and better grasping cameras.

Below are the many points that were considered for the new design, coding, circuiting and operation of the autonomous vehicle.

- The design and dimensions of the steering were chosen to be at level with the motor and to propel the car.
- Cameras were selected that would fit the functioning of this autonomous car so the vehicle can follow a specific object with clear vision, proper signals and object identification.
- The locations of the cameras were selected so that the camera could follow the lead vehicle at all times.
- The motors were selected so they had enough torque to propel carry the weight of the car.
- A microcontroller was selected that would be compatible with the functions necessary to perform.

The vehicle design has three aspects: physical fabrication and modification of the vehicle, circuit component and layout design, and software development for control of the autonomous vehicle. This thesis primarily focuses on the first two.

The autonomous vehicle can be broken into two important systems: object visualization, and vehicle control. Object visualization was handled through the use of two Pixy2 Object Tracking cameras and an ATmega microcontroller. The two cameras work together to provide information that is processed by the microcontroller that can send input to the vehicle control system. The vehicle control system can be broken into two areas: data processing and throttle/steering control. The data processing is handled by the ATmega microcontroller. The microcontroller receives signals from the cameras

microcontroller to make decisions and transmit those decisions to the main vehicle control; throttle/steering. Another added feature is the use of an emergency brake on the car. This is controlled by a servo and has two settings either full brake on or full brake off. Steering control is handled by using servos and a chain driven system to move the steering column.

The three main design objectives for fabrication and modification of the vehicle are:

1. Mounting the dual-camera system to the vehicle.
2. Mounting the stepper motor for brake control to the vehicle.
3. Mounting the stepper motor for steering control to the steering column.

Cameras are mounted on a long, steel square beam bolted to the front of the vehicle. This bar is held fixed through the use of two steel L-brackets that are mounted directly to the vehicle's chassis. These components were chosen because they were manufactured at a local machinist's store and they were purchased at a local vendor. The beam has equally-spaced holes running along its entire length. This is to ensure maximum versatility in the positioning of the cameras. If the spacing between the cameras needs to be adjusted to reduce data errors, the cameras can easily be unbolted and bolted to the beam. All electronic relays and microcontrollers are mounted on a spring-mounted platform attached to the vehicle chassis. The brake servo is directly mounted to the

chassis through the use of a U-bolt. The steering servo is housed in a welded steel square bar that is directly bolted to the chassis.

The vehicle control system decision began with making improvements on the senior design project that initially retrofitted the vehicle. The Arduino circuit and code that were used did not work out move the autonomous car. Also, in the design project, the soldering for the connections were very weak and might would have caused a short circuit. We removed the whole Arduino board and connections. During the initial design phase of the autonomous vehicle, factors relating to fabrication, circuit layout, and software development were taken into consideration. For fabrication, the vehicle had to be modified in such a way that it would be physically able to drive itself; this would include steering, braking and acceleration. The vehicle acceleration was selected to be directly controlled using a microcontroller, while the control of the steering and braking would be through the use of stepper motors controlled by the microcontroller.

For the brake control, a linear actuator was considered due to the high amount of force that needs to be generated and simplicity of mounting. However, a stepper motor was ultimately chosen because of its lower cost and larger scale of available information. Attached to the stepper motor is a spool, which is used for the brake wire to be wrapped around and to be held in place.

To properly use the steering stepper motor, a second steel steering column was fabricated to house the motor. The column was produced such that it is oriented at the same 71 degree angle as the vehicle’s steering column. The motor is held

in place at the top of the steel column by a thin steel bracket which is bolted to the column. Small gears are placed on the steering motor shaft and on the top of the vehicle steering column, and are connected by a miniature chain.

A circuit was developed with the power supply connections, as discussed in *Section 3.3*. The power supply gets connected to the ATmega Micro controller. All the components were soldered on the board as the power supply circuit shows in order. All the connections on the circuit were soldered as shown in the circuit diagram in *Figure 3.1*. *Figure 3.2* shows circuit connections diagram for the cameras.

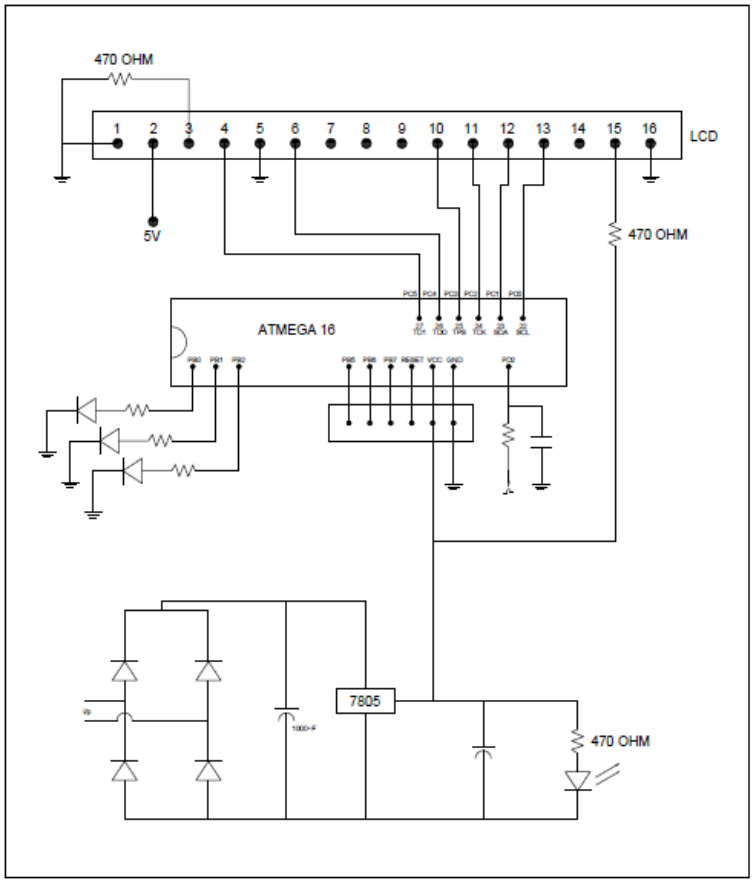


Figure 3.1. Main Circuit Board Connections Diagram.

The steering stepper motor wires were connected to the ATmega microcontroller by four input male pins of the L298D motor driver. The other four wires of the eight-wire stepper motor are connected to the circuit ground connection. Four wires connect the motor driver output pins to D2, D3, D4 and D5 input pins of the ATmega microcontroller which drives the motor through motor controller. The choice of wires to be grounded and the sequence of wire connections (which is D3, D2, D4, D5 – motor moves clockwise (vice versa for counterclockwise) were reached after testing and following the procedure described in *Appendix 4.* These 8-wire motor connections took many iterations as to find the right combination and sequence of steps and connect accordingly.

The brake stepper motor wires were connected to the ATmega microcontroller by four input male pins of the L298D motor driver. The other four wires of the eight wire stepper motor are connected to the circuit ground connection. Four wires connect the motor driver output pins to D6, D7, D8 and D9 input pins of the ATmega microcontroller which drives the motor through motor controller. This choice of wires to be grounded and the sequence of wire connections (which is D7, D6, D8, D9 – motor moves clockwise (vice versa for counterclockwise)) were detected after testing, following the procedure described in *Appendix 4.*

The 12V pin of the motor driver is wired with the 12V output of the circuit power supply, the 5V pin of the motor driver is connected to the 5V output of the power supply and the ground of the motor driver is connected to the ground of the circuit. Two of the special three output pin blocks with these three connections was soldered on the circuit to make connection easier for both stepper motors.

The main DC motor, which propels the vehicle, is connected to the circuit in the following way: The two wires of the DC motor are connected to the COM (common) pins of the relay module. The two NO pins are connected to the positive of the battery and and two NC pins get connected to the negative of the battery. The ground connection of the relay gets connected to the ground of the circuit, VCC pin gets connected to the 5V output of the circuit and the main driving pins of the relay module connects to the pins A2 and A3 of the ATmega microcontroller. The 12V pin of the motor driver is wired with the 12V output of the circuit power supply, the 5V pin of the motor driver is connected to the 5V output of the power supply and the ground of the motor driver gets connected to the ground of the circuit. Two of the special three output pin blocks with these three connections was soldered on the circuit to make connection safer for both the stepper motors.

The 12V battery is connected to the circuit in the following way: The positive terminal of the battery is wired to one end of the main power switch and the other end of the switch is wired to the positive of the circuit power supply. The negative terminal of the battery is connected to the negative terminal of the circuit.

The cameras have pin input manifolds on their back. Pin 1 of the camera is wired to the Pin A0 of the microcontroller and Pin 6/ (GND) of the camera is wired to the ground connection of the circuit. Both cameras are connected to a laptop for power supply and live feed on the laptop, with a USB cable that comes with the camera.

The camera gives a low output of about 2V which was not getting easily detected by the circuit after loading the code and turning on the vehicle. The Pixy cameras give an output signal of 2V. Hence one more connection was needed in the circuit and the code was modified accordingly. The VCC port of ATmega 16 microcontroller was connected to the VCC port of the power supply. The cameras are connected to the laptop through an USB cable and then the cameras are trained through the Pixymon V2 software on the laptop, with interface selection of analog/digital. The different objects that the cameras are trained with are saved in the software and the cameras can be made to follow any of these objects they are trained with, at any time loading the saved object program from the Pixymon V2 software.

The code is written in BASCOM software and compiled as a BASCOM file which is then loaded to ROBOKITS AVR Software. Then, a loader (hardware) is used between the microcontroller and laptop. The code is then uploaded in the microcontroller through the ROBOKITS AVR software and loader.

After all these connections are made and the program is loaded, the main switch is turned on and the autonomous car begins to function on its own following the object it is trained to follow, giving a live feed of its surroundings at each second on the laptop screen. If the object is not detected, a white LED on the circuit starts blinking which indicates that the object is out of range or not getting detected.

The plan was to use PWM Technology, to vary speed of the vehicle. PWM stands for Pulse Width Modulation. For this, first an IR sensor needs to be connected in the main circuit with ATmega16 and placed in front of the car. We can get different speeds by applying different duty cycles to the motor. When we get larger pixel value we decrease our duty cycle, hence decreasing the speed of the vehicle. This way, when the sensor senses the vehicle in front of it in close proximity, it gives higher pixels output which instructs the microcontroller to reduce the speed of the main motor. This command will be integrated in the code in future iterations.

To turn the vehicle the angle from the camera to the change in angle of the lead vehicle is being checked by both the cameras every second and as the motors are step motor, the steps could be increased or decreased to change the speed. When we get signal from the right camera we command the motor to turn the vehicle right until we get some output from the left camera (hence output from both the cameras). We make the vehicle turn in the left direction the same way.

3.2 Code Explanation

All the basic LCD pins are introduced by first specifying the pins of the microcontroller they are connected to.

“$regfile = "m16def.dat"

$crystal = 8000000

$baud = 9600

Config Lcdpin = Pin , E = Portc.4 , Rs = Portc.5 , Db4 = Portc.3 , Db5 = Portc.2 , Db6 = Portc.1 , Db7 = Portc.0

Config Lcd = 16 * 2”

Then the line of code

“ Config Adc = Single , Prescaler = Auto , Reference = Avcc

Start Adc “

is added to the code so the microcontroller can read the low voltage outputs of the camera when it detects the object it is trained to.

Two variables W1 and W2 are introduced, which depict the voltage outputs from the PINs A0 and A1 of the microcontroller which takes the inputs from the cameras. All the output pins of the microcontroller are defined for the microcontroller to identify different parts, by the code “Config”. By this the microcontroller can control the stepper motors, DC motor and LEDs.

A loop begins, and the first line defines W1 and W2 as the voltage outputs of the cameras at the Pins A0 and A1.

```
“Do
W1 = Getadc()
W2 = Getadc(1)
If W1 > 50 And W2 > 50 Then
Gosub Move1
Elseif W1 < 50 And W2 > 50 Then
Gosub Move2
Elseif W1 > 50 And W2 < 50 Then
Gosub Move3
Else
Set Portb.0
Waitms 100
Reset Portb.0
Waitms 100
End If
Loop
End ”
```

The code states that if the voltage output from both cameras is more than 0.5V (when both the cameras detect the object) then the microcontroller is sent a signal to drive the main DC motor forward which moves the car forward.

If the voltage output from the right camera is more than 0.5V and if the voltage output from the left camera is less than 0.5V (when only the right camera detects the object it is trained to see), then the steering stepper motor runs in clockwise direction, turning the vehicle to the right.

If the voltage output from the right camera is less than 0.5V and if the voltage output from the left camera is more than 0.5V (when only the left camera detects the object it is trained for), then the steering stepper motor runs in anticlockwise direction, turning the vehicle to the left.

If none of the cameras give an output voltage more than 0.5V (if none of the cameras detect the lead vehicle), then the vehicle comes to a stop, the white LED connected to the PORT B0 starts flickering, and none of the motors work except the brake stepper motor that stops the vehicle by rotating counterclockwise. Please refer to *APPENDIX 7* for the code.

Chapter 4

Testing and Error Analysis

4.1 Testing

Tests were conducted part by part and then for the entire autonomous car, as the vehicle is larger in size and weight than previous versions, and not everything could be tested all at once from the beginning. First, the circuit that was soldered and made from the scratch was tested by putting in different codes, to check various parts of the circuit board which included the ATmega microcontroller, the power supply circuit, the main power switch, all the LED lights lights on the circuit, the buzzer, the LCD screen, the push button on the circuit, and the three power outputs/connections on the circuit for the motors and the battery. Several different simple codes were written, loaded in the circuit through the code loader, and each of these were checked one by one to see if all of them are working, are connected and functioning properly, and giving the desired output as they are coded. Many of these parts on the circuit board can be read for future developments and features that can be added for the next iterations such as a LCD screen, the buzzer and push button.

The LCD screen can be used to display any kind of messages, outputs, voltages, and many other things. The buzzer can be used as an alarm for emergency brakes or some malfunction or if the circuit is not detecting anything or low voltages and low power, etc. The push button can be used for changing display messages on the LCD, changing and showing various data on the LCD when the

push button is pushed, can even be used to put in morse code for secret messages, etc. All these parts were checked one by one. We noticed that the buzzer was not working because of a connection problem which was remedied by improving the connection. One or two pins of the microcontroller were not working because of wrong soldering. This defect was also corrected by re-soldering the pin connections. The microcontroller was receiving all the codes and signals and was functioning as it is supposed to. Similarly, all of the other parts that were tested were working and functioning properly. The main switch was tested a couple of times to make sure the connections were good.

Next, the cameras were tested. The cameras were trained with many objects to check if the camera LEDs respond to any object we train the cameras with. This part was straightforward and the cameras responded as expected. However, the cameras were not giving any output voltages from their power output pins, as this voltage output pin sends the signal to the microcontroller which performs its function. Customer service representatives of manufacturers of the cameras were contacted to solve this problem. The solution was to change the interface to analog/digital output. As the cameras are primarily made for Arduino circuits, the interface needs to be changed if used for different circuits. Even after doing this the output voltage was around 2V which was very too small for the circuit to detect. As a solution, a connection was added in the circuit. The VCC ports of the ATmega 16 microcontroller and power supply were connected. After this, when the cameras were tested, only one camera succeeded in all the tests and the other camera was still not getting detected by the circuit. The camera

manufacturers were contacted again to solve this problem which led to a solution of replacement of the camera that was not responding. The Pixy2 cameras have two versions, normal Pixy2 cameras and their LEGO version, which does not have the analog/digital input/output. After changing the cameras the two identical cameras functioned as intended and they gave the input signal to the microcontroller to perform the functions it is supposed to. The cameras were then tested with different codes and objects and the output voltages were checked with a Voltmeter and also with LEDs to test if the input signal was received by the microcontroller. All these tests showed that the cameras were functioning as intended.

Next, the stepper motors and the DC motor were tested. The motors were programmed to see if they rotate clockwise and counterclockwise. They were also tested to see if they rotate when they receive a signal or output voltage from the cameras.

The steering gears were checked next. The gears were placed and adjusted at a position and height that would make the chain going around tight enough so that the steering stepper motor can pull the front tires without any slip.

Once all the parts of the autonomous car were tested separately, the car was tested as a whole. The cameras were trained to follow an object. The car was very heavy and hence it was decided to test the car in an enclosed area rather than outside.

The observations that were made from this test and future recommendations are given below:

- The DC motor has a very high torque, power and speed than the stepper motors. Hence the forward motion is a lot faster than the turning motions, which caused the car to not follow the object a couple of times. This problem can be solved by adding a few components to the main circuit and changing the code accordingly that would control the speed of the DC motor. Also the DC motor has to be replaced by the stepper motor or the stepper motor needs to be replaced by a DC motor. All the motors should have to be of the same specifications of torque and power.
- One of the cameras did not produce sufficient voltage a few times during the motion. All connections were checked but the camera settings seemed to have problem.
- The DC motor takes up a lot of power, which heats up all the thin wires as the power flows through the connections of the circuit. This might lead to burning or breaking of wires. Some wires can be replaced by better conducting and thicker wires.
- The brake wire did not have a perfect grasp on the spool connected to the stepper motor. Consequently, it slipped a lot which made the braking system not function at times. A better way to connect the spool and brake wire needs to be developed.

4.2 Error Analysis

The following errors were observed during the development of the autonomous vehicle.

Errors that were corrected:

- As the camera was Arduino-based, the cable provided with it did not work with this new non Arduino-based circuit. A lot of approaches were tried to connect the cameras to the circuit, which included soldering the voltage source of the camera LED to one of the ports on the circuit, which did not work. The manufacturer of the camera was contacted to find out how the camera should be connected to circuit for analog/digital input/output 1/0. The pins were selected according to the manufacturer's instructions.
- The interface between the camera and the circuit that was elected in the beginning was wrong and hence the camera did not give the voltage output as that it should.
- The camera has two versions, one which is the lego version that does not have the analog/digital interface and the other one that has. In the beginning, the lego version was purchased and because of this, the cameras didn't function with this circuit. Hence the cameras had to be replaced with the analog/digital interface.

Errors to be corrected in future:

- The steering motor gear was not set at the right height due to which the steering motor was not able to pull the front tires with a good grip, as the motor gear needs to be at the same height as the steering gear so that the chain is tight to drive the weight of the car.
- A proper connection between the brake stepper motor spool and the brake wire has not been established. The brake wire keeps slipping from the spool due to which the braking operation is as good as it can be.
- As a result of this, the DC motor uses a lot of power, which heats up the thin wires a lot, as all the power flows through all the connections of the circuit. This might lead to burning or breaking of wires. Some wires would have to be replaced by better conducting and thicker wires.

Chapter 5

Conclusions and Future Work

The car was not functioning when the work was started on it by me. Professor Onur Bilgen and Professor John Petrovsky helped out and upon analysis informed that the soldering was not good at all for the car to function. Work on the coding began and solving the soldering and connections problems on the Arduino Board. The solution was making the circuit and coding more and more complex hence a decision was made to make a whole new circuit and to remove everything from the autonomous car and began to make a whole new car with the Go-Kart chassis. The steering gear and placement, circuit placement, and camera positions were all set again. As a mechanical student, a lot of aspects of electrical connections, circuit and soldering were to be learned. After a month and half of 10 iterations and trial-errors, learning how to solder and develop a new circuit with unlimited future feature additions to the circuit, a whole new circuit was soldered and developed. Many parts and features were added to this circuit like buzzer, push button, LEDs, LCD screen, etc. It took many iterations as to have perfect soldering with the right components with their right connections and the way we need them to function, soldered neatly on a circuit board. The advantage of this circuit was that it was cheaper as it was developed from scratch with any parts required in the use of circuit and the connections were self-soldered hence end number of changes could be made. The coding language had to change from Arduino to a not as complex language BASCOM, which was studied during the duration of this research. This language could be

worked on a lot more easily than the Arduino language because of it's non-complex coding. Next the camera, motors, microcontroller and relay module selections were done after a good amount of research and comparisons of different types and which one would fit the best for the function, the weight and the torque required for this car. Each of the components of the circuit were tested with various iterations of codes, which also made future modifications of the car wider. The 4-wire stepper motors were replaced by the 8-wire stepper motors, the connections which were learned and made by soldering. Both the motors were tested along with cameras and had to go through many trial and errors for both the cameras and the motors to work together. First the stepper motor was used for the forward motion to test car at a lower scale first and then move to the DC motor. A lot of trials were made with coding and placement of spools and wires to propel the back tires with a stepper motor without slipping but these trials did not work. Also the steering motor and gear were to be set at an exact level with the steering, in a way that this stepper motor could pull the front wheels and turn the vehicle with the torque they produce. This took many strenuous trials for the steering and turning to work. Then the camera's functioning had to be checked along with the front and back motors and the lead vehicle. This took many iterations to make it work. Many of the local automotive and electrical shops were contacted for each and every problem encountered and a lot of research was done, for a better electrical knowledge for such errors. At the end when every component was checked, and the forward motion was shifted from the stepper motor to the main DC motor. The car was tested for every function it

was developed for and everything checked, but the DC motor's power and voltage was a lot for the wires to handle and hence all the connections would be needed to be replaced by thicker wires in the future.

The reported work involves an iteration on a previously-developed prototype of an autonomous vehicle. Improvements for the autonomous vehicle were achieved by building a functional circuit and the coding was made simpler. A whole new circuit built from scratch was soldered on the vehicle and proved to be very helpful. Better quality cameras were selected that ended up being essential components for smooth functioning of the car. The cameras give live feed of the surroundings all the time which can be viewed on any laptop. The stepper motors need to be replaced by bigger motors and a IR sensor needs to be connected for speed control. All the wires need to be replaced with thicker wires.

There are other improvements that can be implemented in the future. Better placement position of the cameras, a solution for different torques of the steering and the main driving motors, speed control, steering gear height, servos for camera movements and a better braking system will merely improve performance of the vehicle.

The initial application that this project envisions is military because of high risks to personnel. The project can be expanded to other applications where transportation of goods is involved. Around the globe there are many remote areas where goods are transported in and out using road trains where a semi-truck is pulling more than three trailers. Many of these road trains only carry a

specific item across vast distances, therefore many road trains are needed to supply more items for an isolated area. When this project is applied to a truck, it could enable road trains to follow each other to reduce cost and personnel.

This project does not have any size constraints, as it can be integrated into various transportation vehicles such as boats and semi-trucks. The geographical constraints of this project is dependent on the base vehicle, thus iterations of the autonomous car could ideally only be used on smooth paved roads with small degree of road slope. With the switch to a more capable base car, the autonomous vehicle could be used off-road. Because of the exposed sensitive electronic components involved in this project, the vehicle should only be used in dry or damp weather. Any exposure to water would pose a potential short circuit.

For future product development, a global positioning system (GPS) sensor should be incorporated into the electronics section for more accurate information on vehicle location and heading. In conjunction with a gyroscope, this enables accurate measurement of vehicle velocity and acceleration vehicle calculations in different terrain conditions, and also extend the autonomous vehicle abilities. The vehicle would no longer be limited to smooth and flat surfaces, and will be able to function off-road. The gyroscope and GPS is crucial for all-terrain functionality, because the vehicle can extract terrain information from a preloaded map, which are available online, and readjust its controls system to compensate for slippery and high slope conditions, much like the traction control and anti-lock brakes systems in automobiles.

Another feature GPS would add is the ability to back track from a current location to any previous location. This could be accomplished by continuously saving current steering, throttle and brake settings to memory with a location stamp from the GPS. The autonomous vehicle can now automatically go to any pre-saved location after following a lead car only once. This adds a capability for the autonomous vehicle to be independent from the lead car in certain situations such as delivering supplies to locations it has followed the lead car to.

Due to low torque of the stepper motor used to engage the brake mechanism, an immediate upgrade would be to switch to a linear electric actuator. Similar kind of solution can be applied for the steering stepper motor because of its low torque. Unlike other upgrades mentioned earlier, on upside only involves a minor modification to the current autonomous vehicle. However, the challenge with coding remains since the brake torque is not a linear function with respect to the force applied.

Another feature that can be added is voice command, which would need addition of a few additional components to the circuit and coding would be a challenge. This could make the functioning of the autonomous car slightly in our control when needed.

These additional features and improvements will require extensive programming and testing. For the autonomous vehicle to truly function in all terrains, the base vehicle must have these capabilities. All the control algorithms have to be tuned before switching to a bigger base vehicle that has all-terrain capabilities.

APPENDIX A

Razor Force Ground Drifter

This information was gathered from the product description from the local vendor it was bought from.

Figure A.1 shows the photo of The Razor Force Ground Drifter that was used for the research.

Figure A.1. Photo of Razor Force Ground Drifter. This photo was taken from amazon.com from the images of this product.

Product Dimensions: 11.2 x 40.9 x 28.1 inches

Item Weight: 15 pounds

Manufacturer: Razor ASIN: B002S143RK

APPENDIX B

Features of ATMEGA16

All this information was collected from the product brochure of the ATMEGA16 Microcontroller.[4]

FEATURES:

- High-performance, low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
- 131 Powerful Instructions – Most Single-clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 16 MIPS Throughput at 16 MHz
- On-chip 2-cycle Multiplier
- Non-volatile Program and Data Memories

- 16K Bytes of In-System Self-Programmable Flash

  Endurance: 10,000 Write/Erase Cycles
- Optional Boot Code Section with Independent Lock Bits

  In-System Programming by On-chip Boot Program

  True Read-While-Write Operation
- 512 Bytes EEPROM

Endurance: 100,000 Write/Erase Cycles

- 1K Byte Internal SRAM
- Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
- Boundary-scan Capabilities According to the JTAG Standard
- Extensive On-chip Debug Support
- Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

Peripheral Features

- Two 8-bit Timer/Counters with Separate Pre scalars and Compare Modes
- One 16-bit Timer/Counter with Separate Pre scalar, Compare Mode, and CaptureMode.
- Real Time Counter with Separate Oscillator
- Four PWM Channels
- 8-channel, 10-bit ADC
- 8 Single-ended Channels
- 7 Differential Channels in TQFP Package Only
- 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
- Byte-oriented Two-wire Serial Interface
- Programmable Serial USART

- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator

Special Microcontroller Features

- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated RC Oscillator
- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and

  Extended standby.

I/O and Packages

- 32 Programmable I/O Lines
- 40-pin PDIP, 44-lead TQFP, and 44-pad MLF

Operating Voltages

- 2.7 - 5.5V for ATmega16L
- 4.5 - 5.5V for ATmega16

Speed Grades

- 0 - 8 MHz for ATmega16L
- 0 - 16 MHz for ATmega16

APPENDIX C

BASCOM Software

[4]This information was gathered from the ATmega16 Microcontroller manufacturer's brochure.

- **VCC:** Digital supply voltage.
- **GND**: Ground.
- **Port A (PA7.PA0):** Port A serves as the analog input to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). Pull-up resistors are simple fixed value resistors, that are connected between the voltage supply and the particular pin. The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.
- **Port B (PB7.PB0):** Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the

pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

- **Port C (PC7.PC0):** Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5 (TDI), PC3 (TMS) and PC2 (TCK) will be activated even if a reset occurs.
- **Port D (PD7.PD0)**: Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.
- **RESET:** Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.
- **XTAL1:** Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
- **XTAL2:** Output from the inverting Oscillator amplifier.

- **AVCC:** AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.
- **AREF:** AREF is the analog reference pin for the A/D Converter.

This information was gathered from the software brochure and description online on it is official site, www.bascom.com.

SOFTWARE DESCRIPTION:

Software design includes series of flowchart which helps to track the whole program and enable easy programming and error checking. Software tools are as important as hardware tools required to mould a perfect design.

The following are the different tools used for software development.

- Bascom-AVR
- Robokits AVR USB Programmer (STK 500)

BASCOM SOFTWARE:

Bascom Software is used to provide with Bascom development tools for AVR based microcontrollers. With the Bascom tools, one can generate embedded applications for virtually every AVR derivative. The supported microcontrollers

are listed in the Bascom-AVR. *Figure C.1* shows the interface of the BASCOM software.

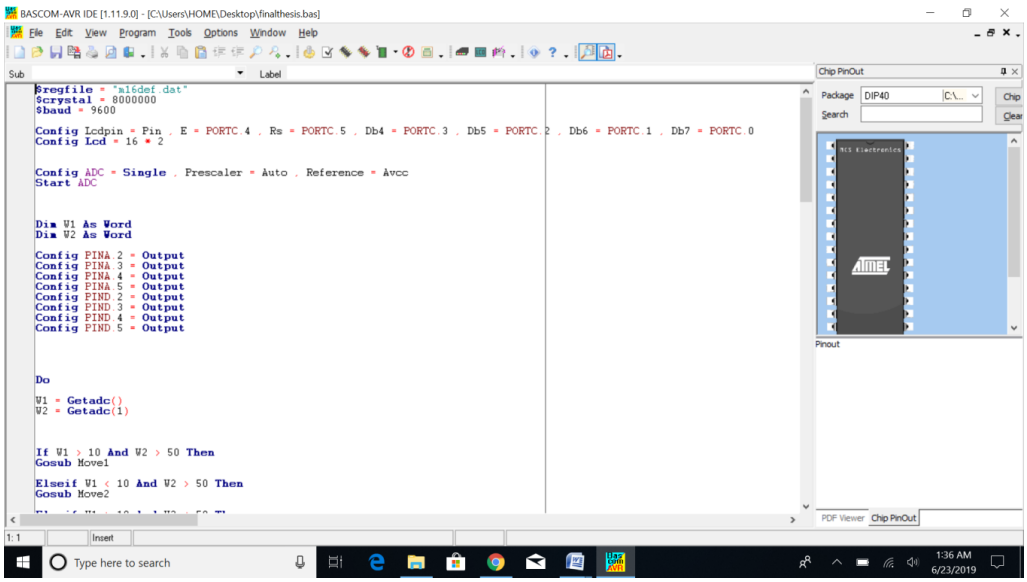


Figure C.1. Screenshot of BASCOM software.

- Since Atmel's AVR microcontrollers are new to the market, they are not as well known as the 8051 controllers. Therefore, this microcontroller family is described here in more detail.

- Atmel's AVR microcontrollers use a RISC architecture which has been developed to take advantage of the semiconductor integration and software capabilities of the 1990's. The resulting microcontrollers offer the highest MIPS/mW capability available in the 8-bit microcontrollers market today.

Figure C.2 shows the BASCOM software startup.

Figure C.2. Software Startup.

- The architecture of the AVR microcontrollers was designed together with C-language experts to ensure that the hardware and software work hand-in-hand to develop a highly efficient, high-performance code.

- The family of AVR microcontrollers includes differently equipped controllers - from a simple 8-pin microcontroller up to a high-end microcontroller with a large internal memory. The Harvard architecture addresses memories up to 8 MB directly. The register file is "dual mapped" and can be addressed as part of the on-chip SRAM, whereby fast context switches are possible.

- All AVR microcontrollers are based on Atmel's low-power nonvolatile CMOS technology. The on-chip in-system programmable (ISP), downloadable flash memory permits devices on the user's circuitboard to be reprogrammed via SPI or with the help of a conventional programming device.

- By combining the efficient architecture with the downloadable flash memory on the same chip, the AVR microcontrollers represent an efficient approach to applications in the "Embedded Controller" market.

Figure C.3 shows program compilation snapshot.

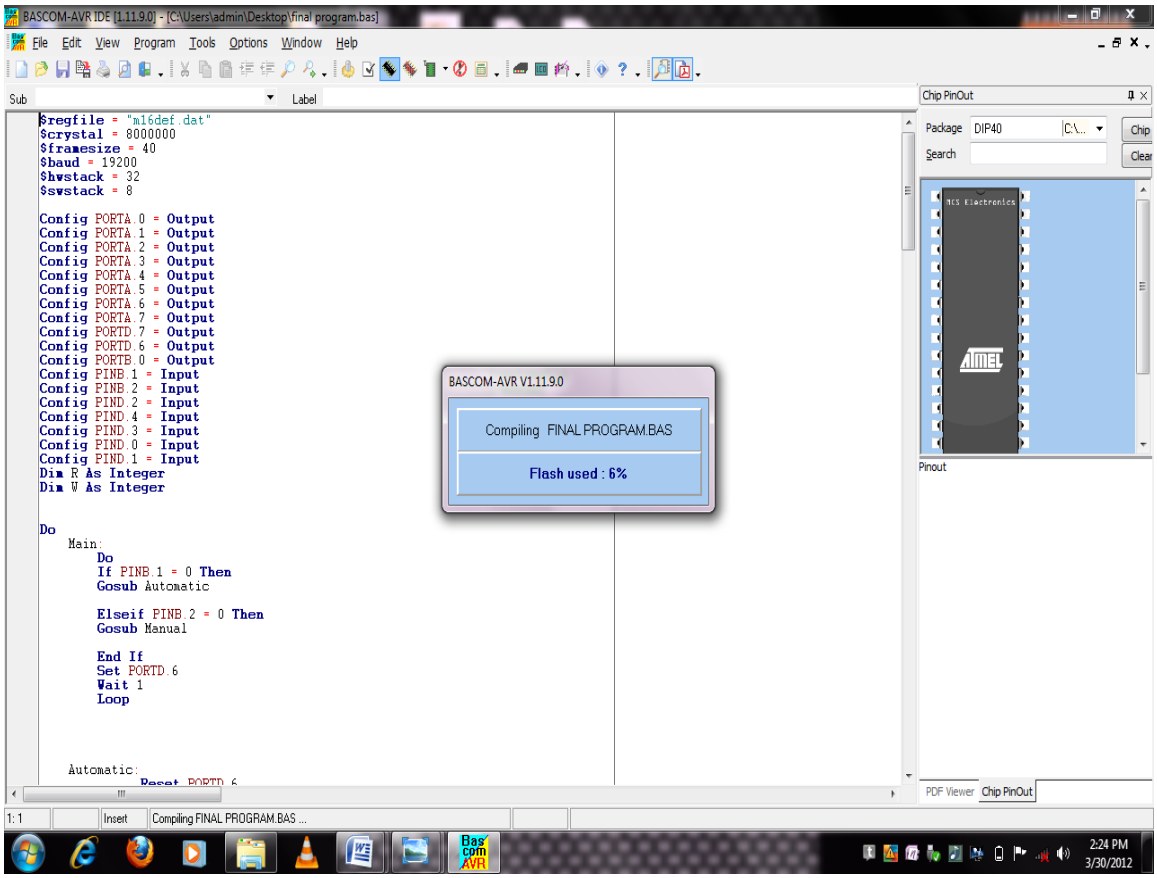


Figure C.3. Program compilation snapshot.

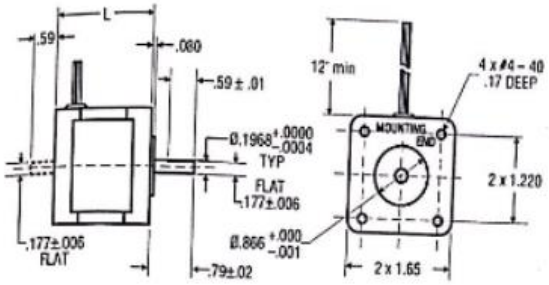
Advantages:

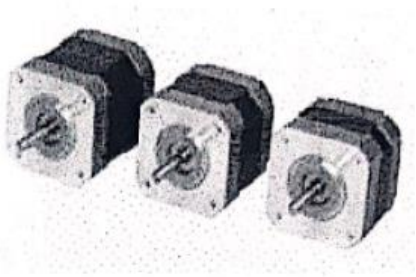
- Easy to interface with different devices.
- Easy to understand.
- Reduces program length.
- Reduces the complexity.
- It is flexible, as it can be worked with other microcontroller by making minimum changes.

APPENDIX D

Stepper Motor selection and connection

[7]*Table 1.*Comparison of various motors in consideration. Taken from www.appliedmotion.com.





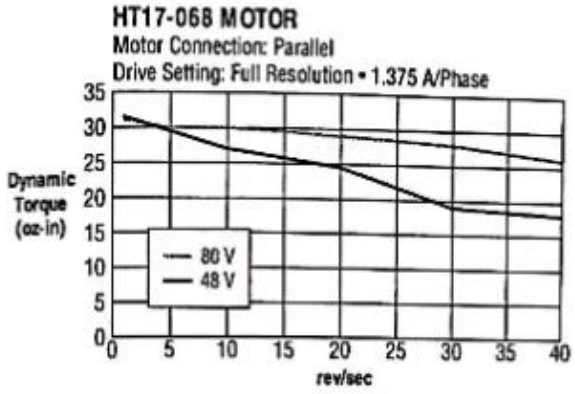
Part #	MOTOR CONNECTION 1 = series 2 = parallel 3 = unipolar	Motor Length (inches)	Minimum Holding Torque (oz-in)	Leads	Step Angle	Volts	Amps	Ohms	mH	Rotor Inertia (oz-in²/G-CM²)	Motor Weight (Lbs.)
HT17-068	1	1.3	31.4	8	1.8	5.7	.67	8.4	11.2	.190/35.0	.44
	2		31.4			2.8	1.34	2.1	2.8		
	3		22.2			4.0	0.95	4.2	2.8		
HT17-069	1		31.4			13.6	0.28	48.0	60.0		
	2		31.4			6.8	0.57	12.0	15.0		
	3		22.2			9.6	0.40	24.0	15.0		
HT17-070	1		31.4			17.5	0.22	80.0	88.0		
	2		31.4			8.8	0.44	20.0	22.0		
	3		22.2			12.4	0.31	40.0	22.0		
HT17-071	1	1.54	51.0			5.7	0.85	6.6	14.4	.29/54.0	.57
	2		51.0			2.8	1.70	1.7	3.6		
	3		36.1			4.0	1.20	3.3	3.6		
HT17-072	1		51.0			9.0	0.57	16.0	30.4		
	2		51.0			4.5	1.13	4.0	7.6		

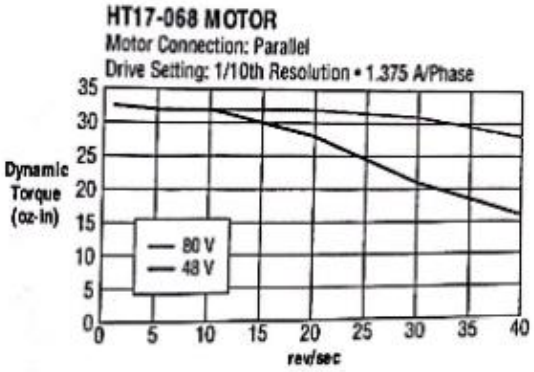
Part #	MOTOR CONNECTION 1 = series 2 = parallel 3 = unipolar	Motor Length (inches)	Minimum Holding Torque (oz-in)	Leads	Step Angle	Volts	Amps	Ohms	mH	Rotor Inertia (oz-in²/G-CM²)	Motor Weight (Lbs.)
HT17-073	1	1.54	51.0	8	1.8	17.0	0.28	60.0	120.0	.29/54.0	.57
	2		51.0			8.5	0.57	15.0	30.0		
	3		36.1			12.0	0.40	30.0	30.0		
HT17-074	1		51.0			33.9	0.14	240.0	424.0		
	2		51.0			17.0	0.28	60.0	106.0		
	3		36.1			24.0	0.20	120.0	106.0		
HT17-075	1	1.85	62.8			5.7	0.85	6.6	12.0	.37/68.0	.73
	2		62.8			2.8	1.70	1.7	3.0		
	3		44.4			4.0	1.20	3.3	3.0		
HT17-076	1		62.8			10.2	0.57	18.0	38.0		
	2		62.8			5.1	1.13	4.5	9.5		
	3		44.4			7.2	0.80	9.0	9.5		
HT17-077	1		62.8			17.0	0.28	60.0	116.0		
	2		62.8			8.5	0.57	15.0	29.0		
	3		44.4			12.0	0.40	30.0	29.0		

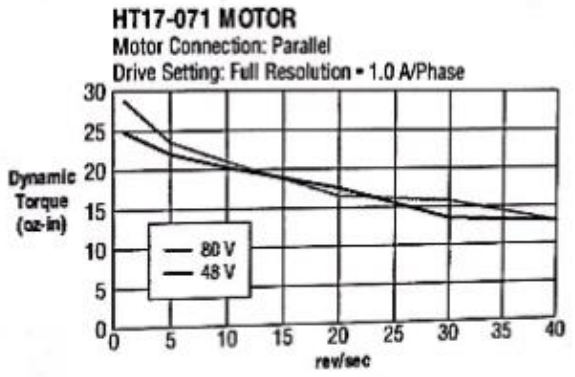
[7]*Table 2.* Size HT-17 Stepper motor typical speed/torque. Taekn from www.appliedmotion.com.

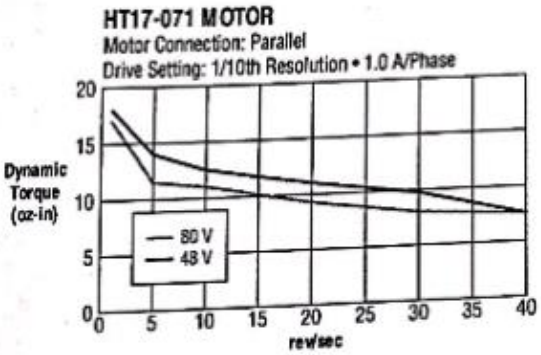
Size HT17 Motor

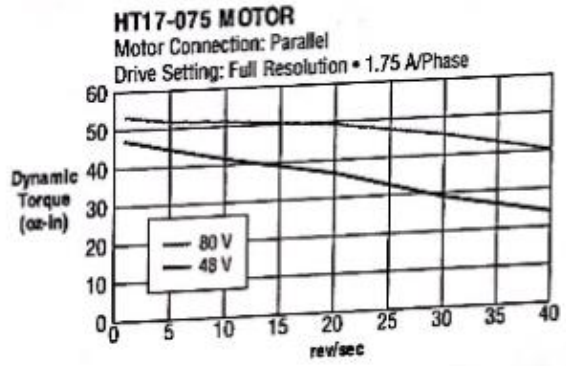
Typical Speed/Torque Performance*

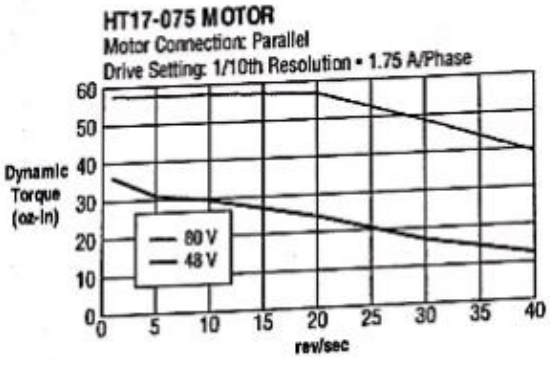












Specifications of HT17-075 Motor:

Part #	MOTOR CONNECTION 1 = series 2 = parallel 3 = unipolar	Motor Length (inches)	Minimum Holding Torque (oz-in)	Leads	Step Angle	Volts	Amps	Ohms	mH	Rotor Inertia (oz-in²/G-CM²)	Motor Weight (Lbs.)
HT17-075	1	1.85	62.8			5.7	0.85	6.6	12.0	.37/68.0	.73
	2		62.8			2.8	1.70	1.7	3.0		
	3		44.4			4.0	1.20	3.3	3.0		

[12]How to connect an eight wire stepper motor(Information taken from www.google.com):

Set up driver with a reasonable current (say 50% of rated value on motor).

- An 8-wire stepper commonly has 4 separate coils. Identify them using a simple ohm meter, to sort out four pairs of cables.

- If not an ohm meter, use one of your driver's output (If your driver has A1 A2 B1 B2, use A1 A2). Randomly plug two out of the eight leads into your driver until the motor shaft vibrates, that two lead belongs together.

  Now suppose you have found your 4 coils, you need to figure out which two belongs to the same pole, and the polarity of the leads. Assuming that the coils are named P Q R S. Where P Q is on one phase, R S on opposite phase.

1. First pick one coil and name the leads P1 P2, this will be connected to you driver's A1 A2.
2. Pick another coil and plug it to the driver's B1 B2, if motor vibrates but not rotate, that coil is on the same phase as coil P, ignore that coil now, try another coil. If the motor rotates, it is the coil on the other phase. If that configuration spin in clockwise, name the leads in B1 as R1, B2 as R2, if it spins in counter clockwise, reverse two leads in B1 B2 and that should make the motor spin clockwise, name them.
3. Repeat that on the other opposite coil, name the leads S1 S2, make sure the motor spins the same way.
4. Next, keep S1 in B1 and S2 in B2 and plug in the remaining unknown coil to driver A1 A2. If motor spins clockwise, name the lead in A1 as Q1, A2 as Q2, otherwise, swap the two unknown wire, that should make the motor spin clockwise, name them.
5. Now you have all 8 leads named, wire them as you wish in a 8 wire, 4 wire (parallel), 4 wire (series) or 6 wire configurations.

Figure D.1 shows photo Stepper Motor HT17-075D that was used.

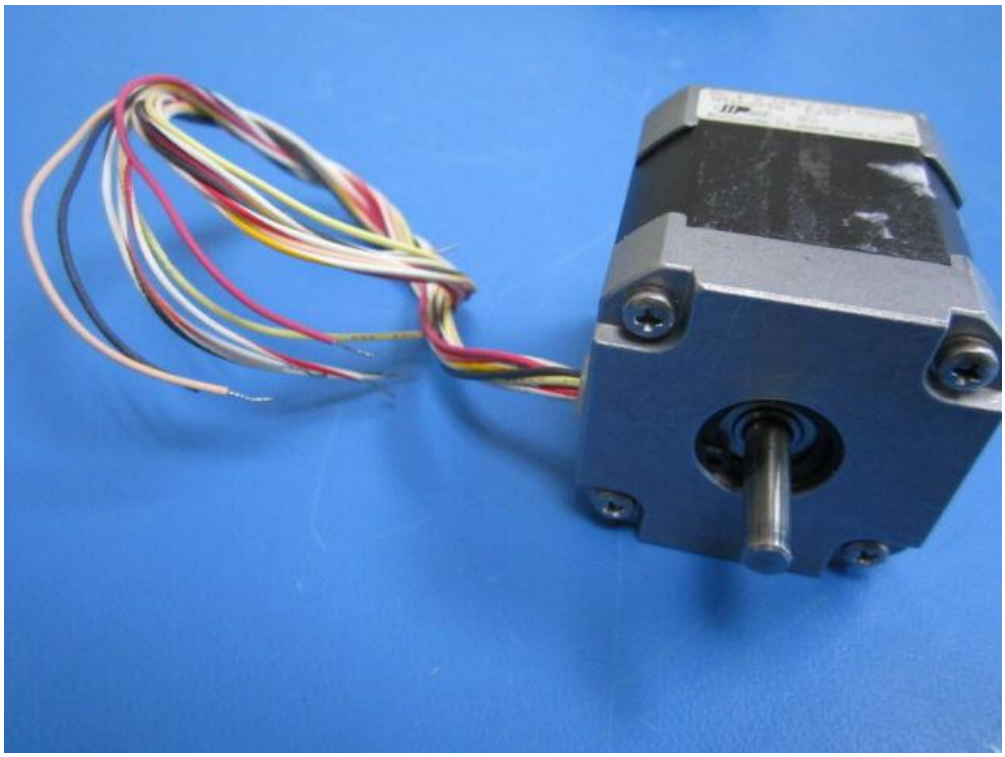


Figure D.1. Stepper Motor HT17-075D.

APPENDIX E

L298N Features and Specifications

Features of L298N Motor Driver and things to remember when using it:

- Dual-channel H-bridge driver working mode creates higher working efficiency, L298N as main chip. The driver can drive one 2-phase stepper motor, one 4-phase stepper motor or two DC motors.
- To avoid damage the voltage stabilizing chip, an external 5V logic supply is needed, especially when using more than 12V driving voltage
- Use large-capacity filter capacitors and diode with freewheeling protection function, increasing reliability
- High working power to 46V, large current can reach 3A MAX and continue current is 2A, power to 25w.
- Large capacity filter capacitance, after flow protection diode, more stable and reliable.

Specification:

- Chip: L298N
- Logic voltage: 5V
- Logic current 0mA-36mA
- Storage Temperature: -20 ℃ to ℃ to +135

- Operating mode: H-bridge driver (dual)
- Drive voltage: 5V-35V
- Drive current: 2A (MAX single bridge)
- Maximum power: 25W
- Dimensions: 43x43x27mm

Figure E.1 shows photo of L298N motor driver that was used.

Figure E.1. L298N Motor Driver.

APPENDIX F

Relay Module Specifications

[3]Qunqi 4pcs 5V 2 Channel 5V Relay Module with Optocoupler Low Level Trigger Expansion Board for Arduino UNO R3 MEGA 2560 1280 DSP ARM PIC AVR STM32 Raspberry Pi

- Driver Current:15-20mA each
- Indication LED's for Relay output status
- IN: can be high or low level control relay
- Equipped with high-current relay, AC250V 10A ; DC30V 10A
- Standard interface that can be controlled directly by microcontroller (Arduino , 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic).

Figure F.1 shows the photo of the Relay Module that was used.

Figure F.1. Relay Module.[5]

Go-Kart DC Motor

Specifications

Model: MY1016 motor by Unite Motor Co. Ltd.

Type: Brush

Voltage: 24 volt DC

Rated Speed: 2600-2850 RPM

Rated Current: 13.5-13.7 amp

Sprocket: 11 tooth #25 chain sprocket

Output: 250 watts

APPENDIX G

Code

```
$regfile = "m16def.dat"
$crystal = 8000000
$baud = 9600
Config Lcdpin = Pin , E = Portc.4 , Rs = Portc.5 , Db4 = Portc.3 , Db5 = Portc.2 ,
Db6 = Portc.1 , Db7 = Portc.0
Config Lcd = 16 * 2
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc
Dim W1 As Word
Dim W2 As Word
Config Pina.2 = Output
Config Pina.3 = Output
Config Pina.4 = Output
Config Pina.5 = Output
Config Pind.2 = Output
Config Pind.3 = Output
Config Pind.4 = Output
Config Pind.5 = Output
Do
W1 = Getadc()
```

```
W2 = Getadc(1)
If W1 > 50 And W2 > 50 Then
Gosub Move1
Elseif W1 < 50 And W2 > 50 Then
Gosub Move2
Elseif W1 > 50 And W2 < 50 Then
Gosub Move3
Else
Set Portb.0
Waitms 100
Reset Portb.0
Waitms 100
End If
Loop
End
Move1:
Set Porta.3
Reset Porta.4
Reset Porta.2
Reset Porta.5
Waitms 30
Set Porta.4
Reset Porta.3
```

```
Reset Porta.2
Reset Porta.5
Waitms 30
Set Porta.2
Reset Porta.3
Reset Porta.4
Reset Porta.5
Waitms 30
Set Porta.5
Reset Porta.3
Reset Porta.4
Reset Porta.2
Waitms 30
Return
Move2:
Set Portd.3
Reset Portd.4
Reset Portd.2
Reset Portd.5
Waitms 30
Set Portd.4
Reset Portd.3
Reset Portd.2
```

```
Reset Portd.5
Waitms 30
Set Portd.2
Reset Portd.3
Reset Portd.4
Reset Portd.5
Waitms 30
Set Portd.5
Reset Portd.3
Reset Portd.4
Reset Portd.2
Waitms 30
Return
Move3:
Set Portd.5
Reset Portd.4
Reset Portd.2
Reset Portd.3
Waitms 30
Set Portd.2
Reset Portd.3
Reset Portd.4
Reset Portd.5
```

```
Waitms 30
Set Portd.4
Reset Portd.3
Reset Portd.2
Reset Portd.5
Waitms 30
Set Portd.3
Reset Portd.5
Reset Portd.4
Reset Portd.2
Waitms 30
Return
```

REFERENCES

[1] Thakur, H., Kodolikar, H. and Padhye, H. (2019). [online] Viva-technology.org. Available at: http://www.viva-technology.org/New/wp-content/uploads/2015/03/ProceedingNCRENB-2015.pdf [Accessed 7 Oct. 2019].

[2] Pixy2 CMUcam5. (n.d.). Retrieved from https://www.sparkfun.com/products/14678.

[3] 4 Channel Relay Module Expansion Board for Arduino: United States. (n.d.). Retrieved from https://salpil.com/tag/4 Channel Relay Module Expansion Board for Arduino/?region=US.

[4] 95808884 Sistem Pencacah Kehadiran Untuk Pengatu Suhu Ruangan Otomatis Berbasis Mikrokontroler ATmega16. (n.d.). Retrieved from https://www.scribd.com/doc/306371168/95808884-Sistem-Pencacah-Kehadiran-Untuk-Pengatu-Suhu-Ruangan-Otomatis-Berbasis-Mikrokontroler-ATmega16.

[5] Basic Relay Schematic - Auto Electrical Wiring Diagram. (n.d.). Retrieved from http://ifsp-srilanka.org/wiring-diagram/Basic-Relay-Schematic.

[6] Bestselling Electric Motor Controls on Amazon. (n.d.). Retrieved from http://gistgear.com/homeimprovement/electrical/electric-motor-controls.

[7] HT17-276 - NEMA 17 High Torque Step Motor. (n.d.). Retrieved from https://www.applied-motion.com/products/stepper-motors/ht17-276.

[8] Wireless Communication – Principles and Practice by Theodore S. Rappaport

[9] Integrated Electronics Analog and Digital Circuits and Systems by Jacob Millman

[10] “1.8 Degree NEMA 17 Stepping Motor Frame 42 mm Stepper Motor 24V, View Stepper Motor, E-TECH Stepper Motor Product Details” Lianglu Transmission Machinery Plant on N.p., n.d. Web. 11 Dec. 2014

[11] “Stepper Library.” Arduino Tutorials. Arduino.cc, n.d. Web. 11 Dec. 2014.

[12] “Stepper Motors.” Code Circuits Construction. N.p., n.d. Web 11 Dec. 2014.

[13] http://www.ridethemtoys.com/sale/11251608/.

[14] https://prodacostore.com/Product/Details/B01HCFJC0Y.
[15]https://www.amazon.com/H-bridge-Controller-DROK-Regulator-Duemilanove/dp/B00CAG6GX2

[16] http://www.dreamationworks.com/topics/myissues/.

[17] http://www.ijaerd.co.in/papers/finished_papers/IJAERDV03I0728685.pdf.

[18]http://www.viva-technology.org/New/wp-content/uploads/2015/03/ProceedingNCRENB-2015.pdf

[19] Final Design Report Spring 2015, Autonomous Vehicle – Dr. Haim Baruh

[20] Critical Design Report Spring 2015, Autonomous Vehicle – Dr. Haim Baruh