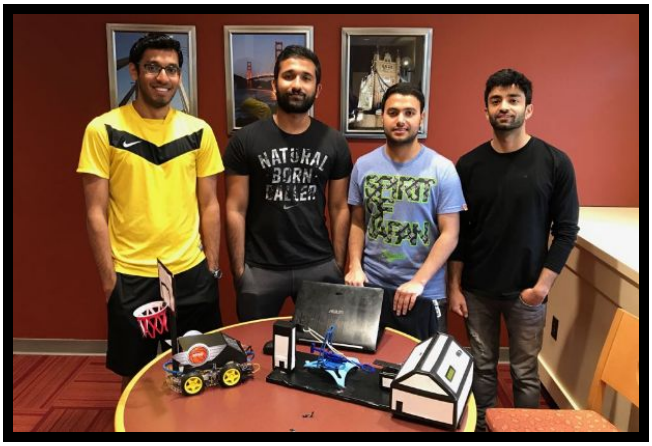
The Mini Hooper Team ID: 11



Team Members (left-to-right on picture, above)	*Class No.*	*Lab Div*
Vaibhav Ramachandran	1044-R	05
Athmaram Swaminathan	2535-S	05
Rochak Chandra	8835-C	01
Milind Shyam	1261-S	03

Report/Functionality Grading Criteria	*Points*
Originality, creativity, level of project difficulty	20
Technical content, succinctness of report	10
Writing style, professionalism, references/citations	10
Project functionality demonstration	20
Overall quality/integration of finished product	10
Effective utilization of microcontroller resources	10
Significance of individual contributions*	20
Bonus Credit Opportunities	*Bonus*
Early completion	0.5%
PCB for interface logic	2%
Poster (required for Design Showcase participation)	1%
Demo video (required for Design Showcase participation)	1%
Design Showcase participation (attendance required)*	1%

**scores assigned to individual team members may vary*

Grading Rubric for all Criteria (Including Bonus)	*Multiplier*
Excellent – among the very best projects/reports completed this semester	1.0 - 1.1
Good – all requirements were amply satisfied	0.8 - 0.9
Average – some areas for improvement, but all basic requirements were satisfied	0.6 - 0.7
Below average – some basic requirements were not satisfied	0.4 - 0.5

Poor – very few of the project requirements were satisfied	0.1 - 0.3

TABLE OF CONTENTS

1.0 Introduction

Our project, the Mini Hooper, makes use of a catapult which is controlled via a microcontroller to launch a ball to a certain distance. The distance the ball travels depends on the amount of tension applied to the springs attached to the catapult's arm. The other end of the spring is attached to the mechanical arm of a servo motor. The amount the servo motor rotates can be adjusted using a dial, which in turn controls how far the spring is pulled. The catapult's arm is held in place by a second servo motor, which is controlled via a pushbutton and is programmed to return to its original position after a certain amount of time. The distance the ball is predicted to travel, is displayed, in centimeters, on an LCD display along with the percentage of the maximum distance it is currently about to travel. The catapult is one half of our project. The other half consists of a basketball hoop attached to a movable base with four wheels. The 4 wheels are attached to 4 DC motors that are controlled using an Arduino microcontroller interfaced using an H-bridge. The movable base is controlled via bluetooth using an Android app. Once the catapult arm (which was previously set) is released, the ball is launched towards the hoop. The hoop can move in all 4 directions - forward, backward, left and right, and its movements are controlled such that the ball may land in the basket.

Role:

- Vaibhav Ramachandran (Software Leader) - Coded the 9S12 microcontroller interface and built the circuit used to control the catapult.
- Athmaram Swaminathan (Team Leader - Packaging and Documentation) - Constructed the basketball hoop, the car hood, designing the catapult mechanism, building the chassis, packaging of all components and documentation.
- Rochak Chandra (Interface Leader) - Developed the Android app used to control the RC car using the bluetooth module.
- Milind Shyam (Peripheral Leader) - Coded the Arduino interface and built the circuit for the RC car.

2.0 Interface Design

The following external interfaces were utilized:

- **Motor Shield** - The motors attached to the cars are powered using a V2.3 Arduino Motor Shield. This is the latest motor shield equipped with a fully dedicated power chip onboard. This chip takes care of all the motors and speed controls over I2C. It was essential to run the motors as it has TB6612 MOSFET drivers, which are alternatives to H-bridges. The drivers enable voltage to be applied across the motors in either direction. Therefore, while switching the polarity of the motors the MOSFETs come in handy, which would otherwise require a bunch of transistors and resistors. The motor shield came with 32 stackable pins, which were soldered onto the shield based on the requirements of the circuit design.

- **Arduino Uno Bluetooth module** - The Arduino uses an HC-06 Bluetooth Module with the default baud-rate of 115200 bps. It had four pins of which two were used to transmit and receive data back and forth the microcontroller, while the other two were connected to power and ground. The supply voltage ranged from 3.3-6V which was economical as we were powering it up using two AAA batteries. This module was a means of wireless connection from the Android application to the Arduino microcontroller.

- **LCD** - The 2x16 LCD is interfaced with the GAL-device which implements the 8-bit shift register. It's used for the following purposes :-
    - When the 9S12C is powered up, the LCD displays a welcome message.
    - Displays the percentage of torque applied.
    - Calculates an approximate distance covered by virtue of the torque applied.
    - Displays an error message with the torque exceeds its maximum limit.

- **Potentiometer** - The potentiometer implements the ATD peripheral of the 9S12C microcontroller. The first and third terminals of the potentiometer are connected to power and ground respectively while the middle terminal provides an input to ATD. The potentiometer used has a 10 kilo-ohm variable resistor with a sliding contact acting as a voltage divider. This is mounted next to the LCD display in order to make it easier for the user to see the amount of torque applied.

- **Pushbutton** - The 2 pushbuttons used are 3 terminal pushbuttons which provide momentary contact closures to ground once pressed. Pins 1 and 3 are connected to power and ground respectively and the middle terminal provides the digital input to the ATD.

- **Servo** - The 2 servos used were Futaba s3003 servos which had 3 inputs, namely power, ground and the signal to control the rotation of the servo. The signal input of the servo is connected to the PWM channel output and receives the signal from the microcontroller which controls its duty cycle and thus its rotation.

3.0 Microcontroller Resource Utilization

Peripherals Utilized:

- SPI - this peripheral was used to interface the LCD display through an 8-bit shift register from a GAL device to the 9S12C microcontroller in order to display the estimated distance the catapult will throw the ball and also display the percentage of torque of the servo.
- ATD - We needed an analog to digital convertor to convert the voltage values provided by the potentiometer into digital numbers which can be understood by the microcontroller. The potentiometer was used to control the rotation of the servo which in turn caused the rubber band to stretch by varying amounts. We also set up the ATD to sample the digital input signal from two pushbuttons which would be used to control the release servo (which holds the catapult arm in place when tension is applied to the rubber band) and to start the program on the microcontroller which rotates the servo to provide tension in the rubber band.
- PWM - this peripheral was used for two different applications:

a. Servo 1 to stretch the rubber band - we perform an ATD conversion on the DC voltage values provided by the potentiometer whose converted digital values are then used to control the duty cycle of the servo motor.
b. Servo 2 to release the ball - a pushbutton is used to trigger the servo motor to rotate whose duty cycle is fixed to a particular value.

- TIM - the timer module was used in servo 2. The timer module was initialized to generate an interrupt every 10ms and each time it generated an interrupt a counter was increased. Once 10 seconds had elapsed the counter would be reset and the duty cycle of the servo motor would change to make it return back to it's original position.

Other on-chip resources used:

- RTI - used to sample the inputs every time the pushbuttons were pressed. Two pushbuttons were utilized: One for starting the microcontroller which controlled the servo pulling the rubber band and the other for controlling the release servo which holds the catapult arm in place when tension is applied to the rubber band.

4.0 Software Narrative

Software for Arduino based RC Car

The car is controlled using an Android application. The application leverages a stream of bytes to communicate to the bluetooth module(HC-06) attached to the car's chassis. A network socket is created which interacts between the user(the application) and the client(HC-06)by opening a bidirectional connection to send and receive a stream of bytes. The application has four buttons, one for each direction, i.e. Forward, Backward, Left and Right. The application has a byte assigned to each direction which is then conveyed to the four motors connected to the car via the bluetooth. The application's layout is made using XML and is configured to API 16 in order to target multiple devices. On the other hand, The car is being controlled by Arduino Uno R3. The arduino, is programmed to co-ordinate and control all the four motors simultaneously for better maneuvering of the car. It establishes a bluetooth connection using a baud rate of 9600 due to the transmission rate of the application which is also close to 9600 bps. The program runs in a loop after the bluetooth is connected in order to get continuous commands from the application. Each command is mapped to a specific control of the motors, i.e. each of the motors are run or stopped based on the command being transmitted from the application. Since its a loop, the user has to indefinitely press the button for the arduino to perform the particular action otherwise it stops.

Software for 9S12C Microcontroller

The catapult is controlled via a 9S12C microcontroller chip interfaced to two servo motors, two pushbuttons, a potentiometer and an LCD display. The software operates in an event-driven fashion, i.e., certain actions only occur once the user has either turned the potentiometer knob or has pressed one of the pushbuttons. Once the program has started, the LCD display turns on but displays nothing until the start pushbutton is pressed. Once the start pushbutton is pressed, a welcome message is displayed on the LCD screen for about 10-15 seconds before the program starts sampling the ATD inputs and the PWM outputs at a rate set by the TIM module. The LCD display now shows the projected distance the ball will travel and the percentage of the max torque the servo is currently applying. Once the sampling of the ATD inputs has begun, the servo responsible for creating tension in the rubber band, rotates depending on the analog signal provided by the potentiometer which is continuously sampled from ATD channel 0. This analog signal is used to determine the duty cycle for channel 3 of the PWM output, which controls the rotation of the servo. The servo stays in that position till such time the signal is changed. The message on the LCD display shows the appropriate calculated distance and the torque percentage depending on the analog signal input. The ATD is also configured to provide 2 digital inputs on channel 1 and 2. These inputs are continuously sampled by the RTI module. The digital signal from channel 2 is the start pushbutton which starts up the entire circuit. The digital input from channel 1 is the servo release pushbutton controlling the rotation of the second servo. If the release pushbutton is pressed, then the release servo turns 90^{o} releasing the catapult arm, and a counter variable is initialized, which counts up to 10 seconds. Once the counter reaches 10 seconds the release servo returns back to it's original position.

5.0 Packaging Design

- **Packaging the Arduino based RC Car -** 4 motors, a 4 wheeled chassis and a plastic base were ordered online. These parts were then assembled using screws and bolts making sure that the connections are made such that polarity of the motors running the wheels are oriented in the same direction. The wires which connect the motors to the wheels were soldered so that there is no disturbance in connection. Cardboard was used to build the hood of the car and black electrical tape was used to color the hood. Plastic case was cut and used to make the windshield of the car. LEDs were used to make the headlights. Finally a logo was designed and pasted on the car with super glue.

- **Designing the Basketball hoop -** Plastic cup was cut and designed as a basketball net. Cardboard was used for the backboard. White paper and electrical tape was used to conceal the cardboard. A bunch of wooden skewers were assembled together and used as the pole of the backboard.

- **Packaging the Catapult circuit -** Cardboard was used to conceal the circuit operating the catapult mechanism. Holes were cut on the top of the package and a Potentiometer, LCD screen and Pushbuttons were inserted onto it. White paper and electric tape was used to put the package in place.

- **Packaging of the Servos -** Cardboard was carved and the servos were placed into it. Electric tape was used to rigidly hold the servos in place. A wooden rectangular piece with a hook attached was mounted onto one of the servos which was used to pull the rubber band of the catapult. Another wooden rectangular piece was mounted onto the other servo strong enough to hold the arm of the catapult in place and withstand tension created by the rubber band which is being stretched by the other servo.

- **Final mounting of all components -** The package for the Catapult circuit, the servos and the Catapult were all mounted onto a sturdy wooden plank using gorilla tape. The hood for the RC car was mounted onto the base carrying the 4 wheels using electrical tape.

6.0 Summary and Conclusions

We learnt quite a lot after completing the project. We understood the functionality and use of all the different on-chip peripherals on the 9S12C32 microcontroller, specifically PWM, ATD, TIM, SPI and RTI. We used PWM to control the rotation of the servo motors attached to the catapult. The potentiometer utilized the ATD module to convert the analog voltage supplied into a digital number recognized by the microcontroller which was used in rotating the servo motor that pulled the rubber band. We used the TIM module to reset the second servo motor which holds the catapult's arm in place and provides tension. The SPI peripheral was used to display the percentage of tension on the LCD. And lastly, RTI was used to sample inputs whenever the pushbuttons were pressed.

We also learnt a lot about the Arduino Uno microcontroller and the bluetooth module, which we used for our RC car with the hoop attached to it. This was the first time we worked with the Arduino microcontroller and its bluetooth module and learnt how to interface the Arduino to the four motors of the car using the motor shield, and also learnt how to control the arduino using the bluetooth module. We developed an Android app to interface with the Arduino Uno over bluetooth.

Our original idea was to implement an automated car with the hoop attached to it that would predict the final location of the ball after it gets launched from the catapult, and then move the car accordingly to catch it. We couldn't implement it due to time constraints and because of the difficulty level. We weren't sure what sensors to use and how to track the ball's location at every point in order to predict its final location. We had a few ideas to implement it using motion sensors, but was not feasible due to the limited time we had.

So, finally we decided to use an app controlled RC car, instead of the automated car, to catch the ball. With this implementation, one person would have to launch the ball from the catapult and the other person would have to try and catch it using the RC car. This way we made our project more interactive and unique.

7.0 References

- "Arduino Controlled Catapult". *Instructables.com*. N.p., 2017. Web. 17 Apr. 2017.
- "Rc Controlled Card For Beginners". *Instructables.com*. N.p., 2017. Web. 20 Apr. 2017.
- "Create An Android App To Control LED's". *Instructables.com*. N.p., 2017. Web. 1 May 2017.
- "View.Onclicklistener | Android Developers". *Developer.android.com*. N.p., 2017. Web. 23 Apr. 2017.
- Industries, Adafruit. "Adafruit Motor/Stepper/Servo Shield For Arduino V2 Kit [V2.3] ID: 1438 - $19.95 : Adafruit Industries, Unique & Fun DIY Electronics And Kits". *Adafruit.com*. N.p., 2017. Web. 24 Apr. 2017.
- "Bluetooth Module (Device)". *Hobbyist.co.nz*. N.p., 2017. Web. 24 Apr. 2017.
- “Bluetooth Controlled Arduino RC Car”. Instructables.com. N.p., 2017. Web. 23 April. 2017.
- *Instructables.com*. N.p., 2017. Web. 25 Apr. 2017.
- "Bluetoothadapter | Android Developers". *Developer.android.com*. N.p., 2017. Web. 21 Apr. 2017.
- "Bluetoothsocket | Android Developers". *Developer.android.com*. N.p., 2017. Web. 20 Apr. 2017.
- "H Bridge". *En.wikipedia.org*. N.p., 2017. Web. 27 Apr. 2017.
- H. E. C., “Arduino Cotrolled Catapult,” *YouTube*, 27-Apr-2012. [Online]. Available: https://www.youtube.com/watch?v=R8D8ZZJyCLA. [Accessed: 01-May-2017].
- Y. Söylemez, “Car and Motorcycle Logos,” *Pinterest*, 16-Mar-2012. [Online]. Available: https://www.pinterest.com/pin/268456827757827257/. [Accessed: 01-May-2017].

Appendix A:

Individual Contributions and Activity Logs

ECE 362 *Microprocessor System Design and Interfacing* Spring 2017

Activity Log for: Vaibhav Ramachandran **Role:** Software/Peripheral Leader

Activity	*Date*	*Start Time*	*End Time*	*Time Spent*
Coding the 9s12c32 microcontroller. Initializing peripherals.	4/21/2017	6:00pm	8:00pm	2 hours
Researching about servo interfacing and programming. Method to interface 2 servos to the same microcontroller, without signal interference to either one. Microcontroller programming. Debugging software.	4/22/2017	12:00pm	10:00pm	10 hours
Constructing and debugging circuit for microcontroller to be used in the catapult. Interfacing LCD display, with GAL device, pushbuttons and potentiometer and checking basic functionality.	4/23/2017	6:00pm	9:00pm	3 hours
Interfacing servos with microcontroller. Implementing amplifier circuit to provide sufficient power for servos.	4/23/2017	1:00pm	3:00pm	2 hours
Adding push button functionality to control release servo. Use potentiometer to provide input to control main catapult servo's duty cycle.	4/23/2017	5:00pm	8:00pm	3 hours
Implementing further functionality by adding a start pushbutton to start the program in a cleaner, simpler manner. Researched online and performed calculations on relation between torque, velocity and maximum possible range attainable by catapult. Debugging code.	4/25/2017	2:30pm	8:30pm	6 hours
Add in error messages if a certain threshold is reached since the servo is unable to handle a load beyond a certain weight.	4/27/2017	6:30pm	7:00pm	0.5 hours
Final checks on software to ensure smooth operation.	4/27/2017	7:00pm	8:00pm	1 hour
Designing schematic for 9s12c microcontroller circuit	4/28/2017	10:00am	12:00pm	2 hours
Designing poster for Spark Challenge	4/28/2017	12:00pm	2:30pm	2.5 hours
Documentation - Writing of report	4/29/2017	3:00pm	7:00pm	4 hours
Designing poster for online submission	4/30/2017	1:00pm	4:00pm	3 hours
Documentation - Updating report, filling in all necessary sections	4/30/2017	8:00pm	2:00am	6 hours

Written Summary of Technical Contributions: Vaibhav Ramachandran

My role as the software leader was to develop and test the software for the 9s12c microcontroller. Also as the joint peripheral leader, my responsibilities included properly interfacing all the components to the microcontroller such as the LCD display, the pushbuttons, the servo motors and the potentiometer. Another responsibility included developing the schematic for the wiring of the microcontroller circuit as well as the Arduino circuit.
My technical contributions included the following:

- Researching servo control, power requirements and interface mechanisms with a microprocessor. Researched control of the duty cycle of a servo using a potentiometer.
- Coded, commented and debugged the entire 9s12c microcontroller software.
- Created code for the servo's rotation and duty cycle to be controlled using the ATD converted signal given out by channel 0 on the ATD.
- Built the circuit interfacing it to the requisite peripherals and components.
- Debugged the circuit which interfaced 2 servos to one microcontroller by making use of amplifier transistors that would supply sufficient power to the servos.
- Implemented added functionality to code by adding pushbuttons for starting up the set-up.
- Performed calculations to determine the expected range of the catapult at a specific level of torque that the servo motor provides.
- Assisted in debugging of Arduino code for the movable basketball hoop.
- Designed the schematics of both the circuit for the 9s12c microcontroller as well as the Arduino circuit.
- Assisted in the design of the poster for the spark challenge.
- Designed the poster for online submission.
- Documentation of work for final report.

ECE 362 *Microprocessor System Design and Interfacing* Spring 2017

Activity Log for: Athmaram Swaminathan **Role:** Team Leader (Packaging and Documentation)

Activity	***Date***	***Start Time***	***End Time***	***Time Spent***
Assembling the base for the arduino car (4 wheels, 4 motors and plastic base, ensuring correct polarity of the motors).	22/04/2017	4:00 pm	6:00 pm	2 hour
Assisting Software Leader in understanding, writing and debugging code for controlling the rotation of the servo (duty cycle) which stretches the rubber band. Providing ideas in improving functionality of the 9S21C.	22/04/2017-22/04/2017	11:00 pm/6:00 pm	5:00 pm/12:00 pm	6 hours /6 hours
Assisted Interface leader in setting up the circuit involving 9S12C microcontrollers for controlling the 2 servos, LCD display, potentiometer and pushbuttons.	23/04/2017	6:00 pm	9:00 pm	3 hours
Debugging code for Arduino controlling the motors of the car and setting up an initial circuit for the car.	24/04/2017	12:00 pm	6:00 pm	6 hours
Building the basketball hoop and packaging for the 2 servos.	24/04/2017	6:00 pm	12:00 pm	6 hours
Packaging the circuit controlling the catapult.	25/04/2017-	12:00 pm	4:00 pm	4 hours
Mounting all apparatus on a rigid base and hiding all wiring in the circuit.	25/04/2017	5:00 pm	8:00 pm	3 hours
Building the hood of the car and making sure all team members have completed their tasks.	27/04/2017	12:00 pm	5:00 pm	5 hours
Making the poster for the Spark Challenge.	28/04/2017	12:00 pm	2:30 pm	2.5 hours
Making Youtube Video	29/04/2017	7:00 pm	10:00 pm	3 hours
Documentation	29/04/2017	3:00 pm/10:00 pm	7:00 pm/1:00 pm	7 hours
Finalizing Report to be Submitted	1/05/2017	1;00 am	3:00 am	2 hours

Written Summary of Technical Contributions: Athmaram Swaminathan

My responsibilities as a Team Leader was to coordinate with team members, delegate tasks, package all the components and finally create the documentation for our work.

My contributions included the following:

- Brainstorming, researching and debugging code for the control of the servos using our 9S12C microcontroller and assisted in interfacing the circuit involving the 9S12C microcontrollers controlling the catapult.
- Assisting Software and Interface leaders with debugging code and interfacing of Arduino based Car.
- Built the initial test circuit for Arduino based car.
- Providing critical reviews on improving and adding more functionality to the microcontrollers (eg. Using pushbuttons to start the microcontroller).
- Building the Chassis for the Bluetooth controlled Arduino-Car from scratch.
- Packaging of all circuits and components in a neat fashion making sure no wires were on display (includes carving, designing logo, boxing circuits, making car hood).
- Building the Basketball Hoop and attaching to the chassis.
- Bringing together all the final components packaged and attaching them to a rigid base, and testing and measuring the range of the Catapult's throw to be used in the LCD display.
- Making the poster for the Spark Challenge.
- Making Youtube Video.
- Documentation of all work.

ECE 362 *Microprocessor System Design and Interfacing* Spring 2017

Activity Log for: Rochak Chandra **Role:** Software/Interface Leader

Activity	***Date***	***Start Time***	***End Time***	***Time Spent***
Information Gathering- Read about network sockets to create a communication stream between arduino and Android Application.	04/20/ 2017- 04/21/ 2017	10:00 a.m.	3:00 p.m.	5-8 hours
Installed all the necessary drivers pertaining to the project and configured the devices accordingly to make them compatible for the application.	04/22/ 2017- 04/22/ 2017	10:00 a.m.	12:00 p.m.	2-3 hours
Created a multi-platform layout for the Android application using Android XML.	04/22/ 2017 - 04/23/ 2017	4:00 p.m.	8:00 p.m.	6-7 hours
Started working on the back-end of the Android Application, i.e. MainActivity Function.	04/23/ 2017- 04/25/ 2017	10:00 p.m.	1:00 a.m.	15-20 hours
Read about multiple action-driven functions that respond on an event-click. This was necessary as the Android Application was based on the functionality of the buttons.	04/23/ 2017 - 04/23/ 2017	5:00 p.m.	8:00 p.m.	2.5-3 hours
Debugging the software and the circuit.	04/26/ 2017- 04/28/ 2017	9:00 p.m.	12:00 a.m.	15 hours
Soldering the stacking pins on the Arduino Power Shield to Arduino UNO and the wires to each of the four motors.	04/27/ 2017- 04/27/ 2017	8:00 p.m.	9:00 p.m.	1 hour
Editing Project Document	04/29/ 2017	8:00 p.m.	11:00 p.m.	3 hours

Written Summary of Technical Contributions: Rochak Chandra

As a joint software leader I was responsible to create the Android Application which controls the car. The android project was created using Android Studio. The project used API 16 as the target level in order to make the application more compatible with multiple Android devices. Starting with the layout, I used Android XML to design the application. The MainActivity function was coded in Java and imported event-driven libraries to implement on-click functions. The main problem encountered in the app's creation was to create a bidirectional network socket for listening and responding to user's and client's requests. After this was achieved, I executed functionality of all the directional buttons leveraging the onClick event libraries. To debug the app, I also had to establish a serial USB connection from the Arduino to the computer and rectify the errors in incoming transmissions. In addition to the Android application, I also helped in syncing the arduino code with the app. I was responsible for interfacing the RC car's motor connections to the Arduino Microcontroller. This was achieved by soldering the wires of all the motors to respective pins of the controller. This was done in order to prevent any potential loose connections in the circuit due to the speed of the car. The stacking pins were also soldered carefully to the motor shield. The Vin pin of the shield was unsoldered as the driving powers of the PCB's were different. The layout of the circuit connections were inspired from the 'RC controlled car' website.

ECE 362 *Microprocessor System Design and Interfacing* Spring 2017

Activity Log for: Milind Shyam **Role:** Interface/Peripheral leader

Activity	***Date***	***Start Time***	***End Time***	***Time Spent***
Downloaded Arduino coding application and installed drivers.	4/22/17	6 pm	7 pm	1 hr
Downloaded Arduino and Motor Shield libraries.	4/23/17	6 pm	7 pm	1 hr
Researched about Arduino and Motor Shield	4/23/17	8 pm	10 pm	2 hrs
Started coding Arduino Uno	4/24/17	7 pm	11 pm	4 hrs
Debugging	4/25/17	4 pm	11 pm	7 hrs
Built the circuit for the car using Qunqi motor shield	4/25/17	8 pm	10 pm	2 hrs
Changed Arduino libraries and Arduino code compatible for Adafruit Motor Shield. Ordered Adafruit Motor Shield	4/26/17	4 pm	10 pm	6 hrs
Built new circuit using Adafruit Motor Shield instead of Qunqi Motor Shield	4/27/17	3 pm	7 pm	4 hrs
Helped with soldering Motor Shield and DC motors	4/27/17	8 pm	9 pm	1 hr
Worked on Project Document	4/29/17	3 pm	5 pm	2 hrs
Worked on Project Document	4/30/17	10 pm	12 pm	2 hrs

Written Summary of Technical Contributions: Milind Shyam

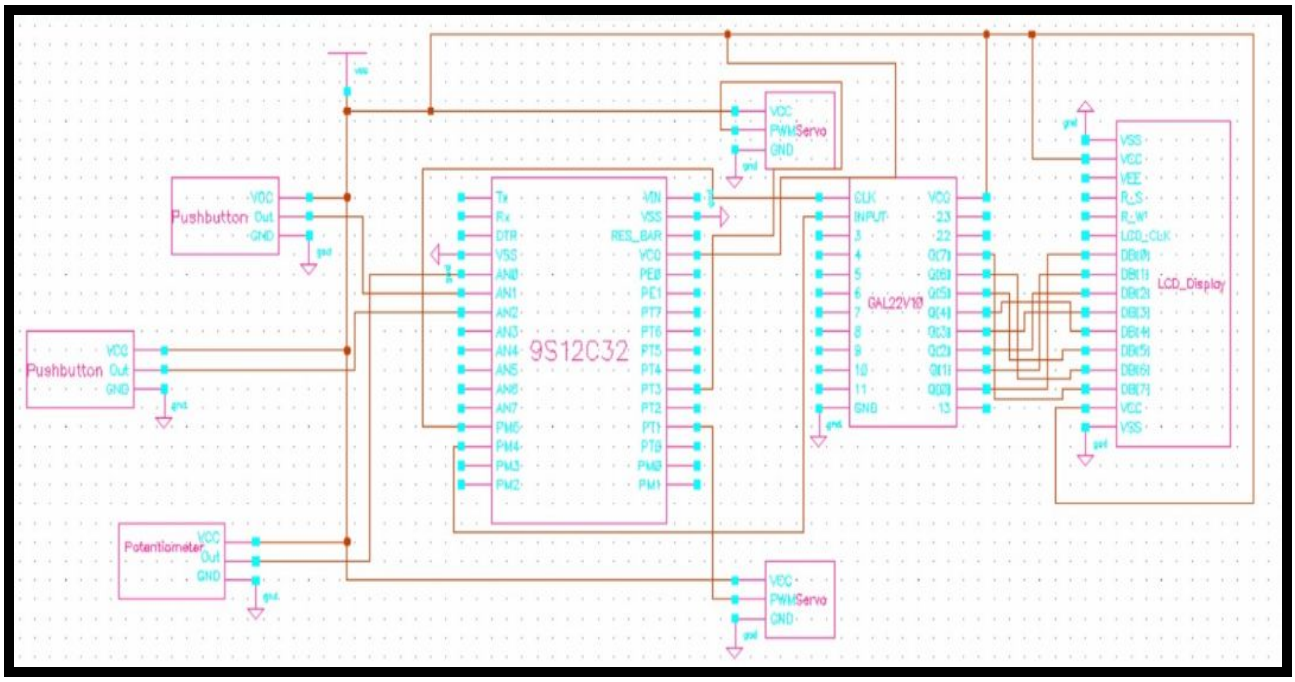
I was responsible for building the RC car's circuit and for programming the Arduino in order to interface it with the Android app via the bluetooth module. I first built the circuit using the Arduino, bluetooth module, 4 DC motors attached to a chassis, and Qunqi motor shield. I then coded the Arduino for the same circuit, but it so turned out that I couldn't find the right libraries for the Qunqi motor shield and the motors wouldn't rotate with any other library. So we then ordered Adafruit Motor Shield. I downloaded and installed the library for the new motor shield and re-programmed the Arduino. I built a new circuit using the Adafruit Motor Shield and tested it and it worked. I also helped with generating ideas for the catapult's functionality and helped solder the motor shield and DC motors.

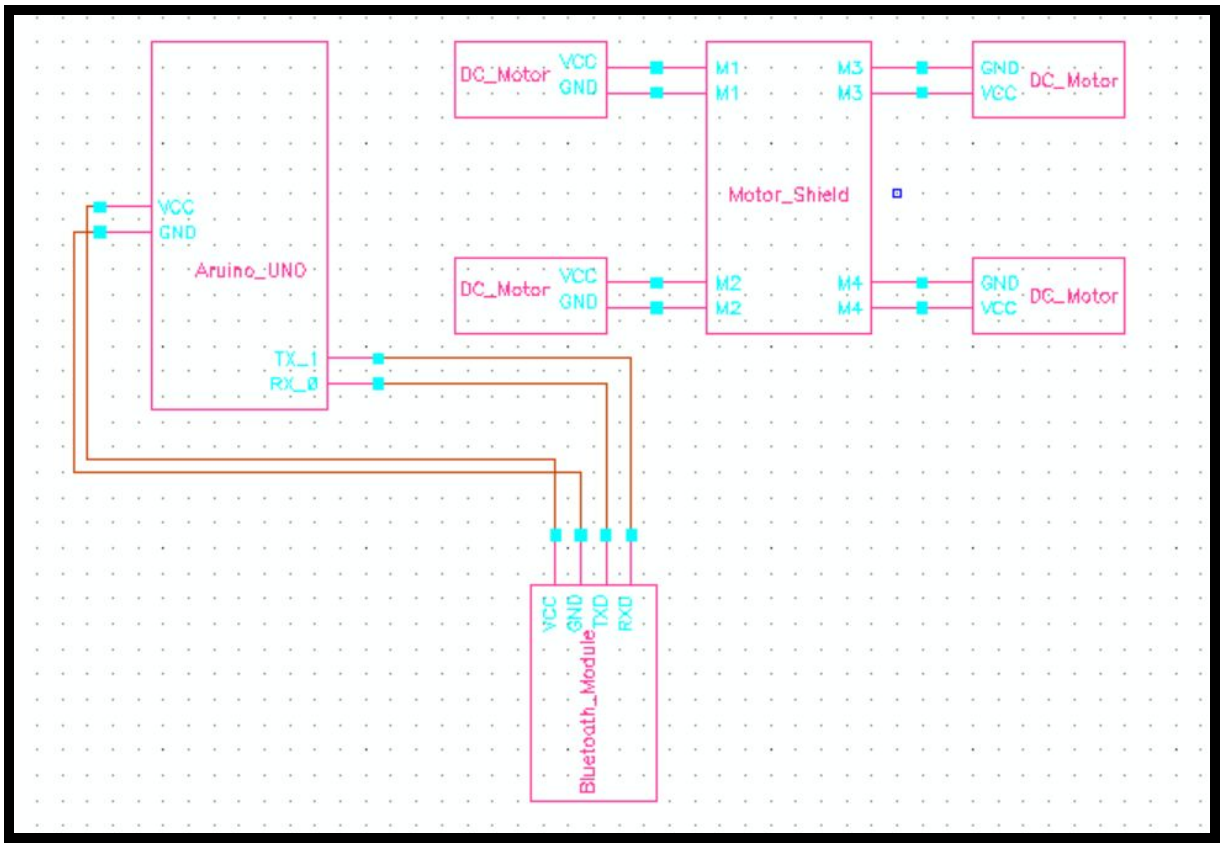
Appendix B:

Interface Schematic and PCB Layout Design

ECE 362 *Microprocessor System Design and Interfacing* Spring 2017

Schematics created by: Vaibhav Ramachandran

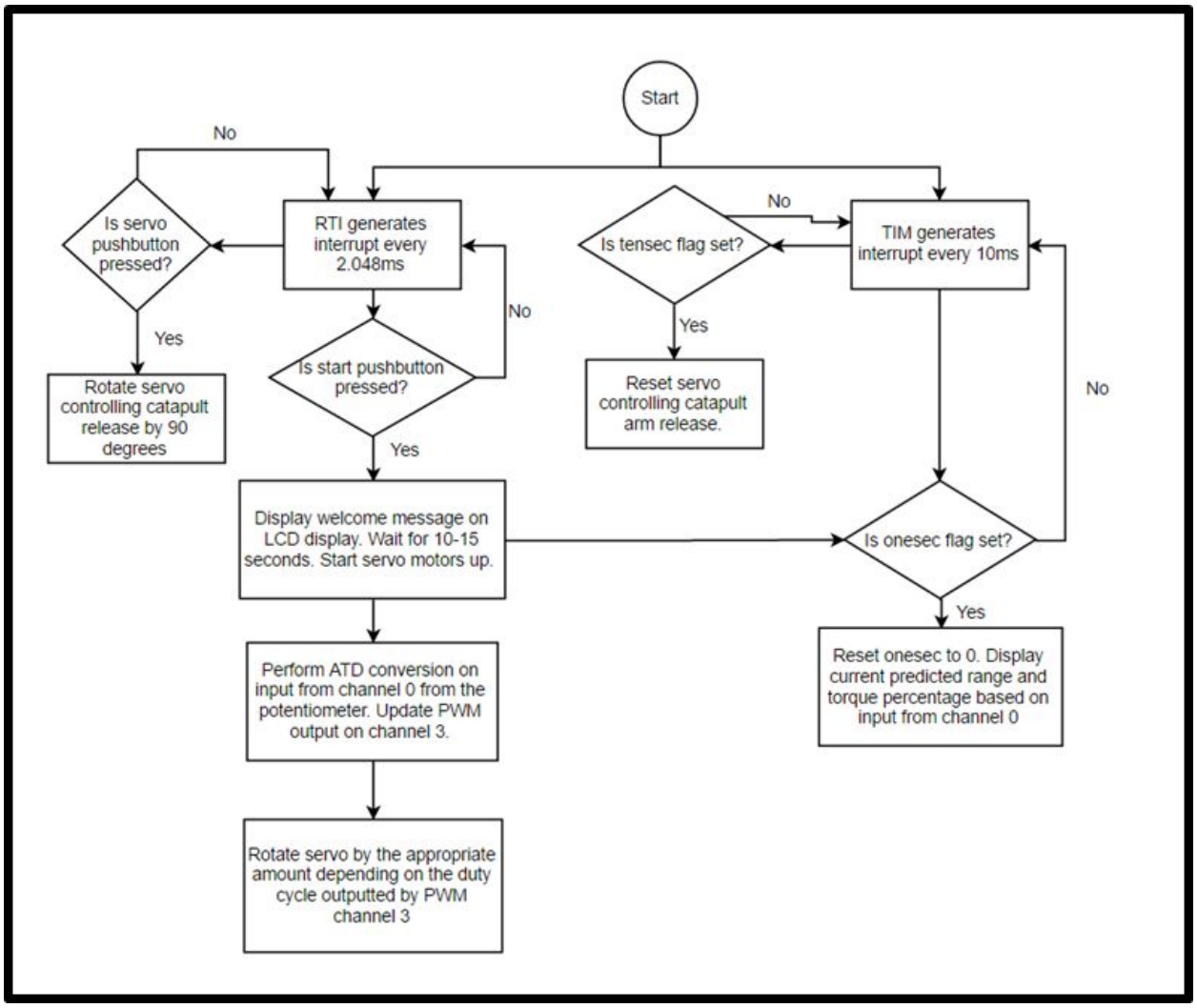




Appendix C:

Software Flowcharts

Flowchart for 9s12c Microcontroller code - Vaibhav Ramachandran



Pseudo-Code for the Android Application - Rochak Chandra

```
public class MainActivity{
        //Initialize all the variables
        Textview
        Button
        Bluetooth Adapter
        Socket streams
        public onCreate() {// This is the first function to be executed when the activity is
launched
                //Map the widgets from the layout to main function
                Bluetooth = findViewById(R.id.open);
                myLabel = findViewById(R.id.label);
                //Implementing listener methods for event-driven actions
                IF the bluetooth is clicked
                        Call the find and open bluetooth functions
                IF the forward button is clicked
```

```
                Call the front() function on pressing and halt() function otherwise
            IF the reverse button is clicked
                Call the rev() function on pressing and halt() function otherwise
            IF the left button is clicked
                Call the move_left() function on pressing and halt() function otherwise
            IF the right button is clicked
                Call the move_right() function on pressing and halt() function otherwise
        }
        void findBt() {
            Initialize the Bluetooth Adapter
            Search for the bluetooth Device(HC-06)
            If found then add the devices to the Set of PairedDevices
            Else create a Toast displaying 'Device Not Found'
        }
        void openBt() {
            Connect the Bluetooth when device is found be using the standard UUID(Unique
Identification Number)
            Open Socket Stream, i.e. both input socket and output socket
        }
        void front() {
            Append 'f' to the output stream message which is transmitted to the bluetooth
        device.
        }
        void rev() {
            Append 'b' to the output stream message which is transmitted to the bluetooth
        device.
        }



        void move_left() {
            Append 'l' to the output stream message which is transmitted to the bluetooth
        device.
        }
        void move_right() {
            Append 'r' to the output stream message which is transmitted to the bluetooth
        device.
        }
}//End of main function
```

Pseudo-Code for the Arduino Motor Shield - Milind Shyam

reference - instructables.com
#include Adafruit_MotorShield library
#include Wire.h library
#include utility/Adafruit_MS_PWMServoDriver.h library
#include SoftwareSerial.h library

initialize bluetooth module's Tx to pin 0

initialize bluetooth module's Rx to pin 1

```
SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

Route the back, front, left and right motors to the appropriate pins of the motor shield-
Adafruit_DCMotor *backMotor = AFMS.getMotor(1);
Route front, left and right motors accordingly to pins 2, 3 and 4.

void setup(){
//Setup initial conditions
Serial.begin(9600); //Set baud rate to 9600
bluetooth.begin(115200); //Set bluetooth to 115200 bps
bluetooth.print("$$$"); //Enter command mode by printing '$$$'
delay(100; //)Set delay to 100
bluetooth.println("U,9600,N"); //Temporarily set bluetooth baud rate to 9600 to relay data
bluetooth.begin(9600); //Start bluetooth serial at 9600 baud rate

AFMS.begin();  //begin using default frequency of 1.6kHz

//Set all motors to start and set speed to 255 (max)
backMotor->setSpeed(255);
backMotor->run(FORWARD);  //turns on motor
backMotor->(RELEASE);  //stops motor

//Set speed accordingly for front, left and right motors as well

void loop(){
//Read data from bluetooth and write data to usb serial
if bluetooth.available()
{
        read character sent from bluetooth

        if character = b
        {
                set back, front and right motors to run forward
                set left motor to run backward
        }
        if character = f
        {
                set back, front and right motors to run backward
                set left motor to run forward
        }
        if character = l
        {
                set front and back motors to run backward
        }
        if character = r
```

```
        {
                set right motor to run backward
                set left motor to run forward
        }
        if character = s
        {
                stop all motors
        }
}
read data from usb serial to bluetooth
}
}
```

Appendix D:

Packaging Design

ECE 362 *Microprocessor System Design and Interfacing* Spring 2017

Created by: Athmaram Swaminathan

- **Car Packaging**

Side View

Front View

- **Basketball Hoop**

- **Catapult Circuit Box**

- **Servos controlling the catapult**