ECE 511

Professor : Dr. Jens Peter Kaps

A project report on

HANDY CAR

GEORGE MASON UNIVERSITY

By :
Arvin Asokan
Nikhil Golani
Sai Kasyap Kamaraju
Urvi Tank

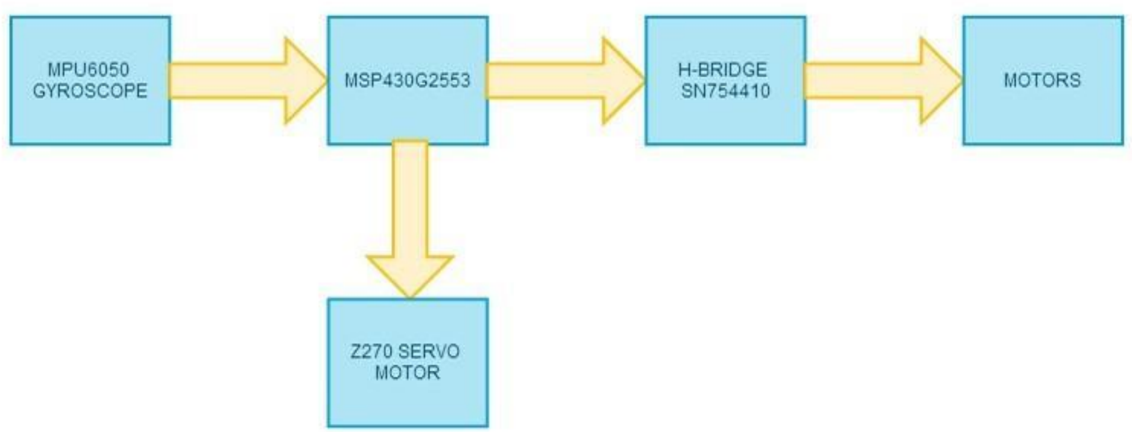
Table of Contents

1) ABSTRACT

Motion Sensing has become a very important aspect these days. The use of motion sensors has become ubiquitous. Motion sensing applications vary from intelligent systems, automation systems, gaming consoles, fitness tracking systems, toys and the list almost seems to be endless. Humans had always been using hand gestures to communicate and at times the use of hand gestures seems to be more supportive when a person tries to express his thoughts. These ideologies gave a spark during one of our brainstorming sessions during the infancy of the project. As the theme of this project was to build an interactive toy, we decided to build a toy which interprets our hand gestures and responds accordingly. After a few brainstorming sessions, we were able to come up with this idea of maneuvering a car using only hand gestures.

The user is given a glove to which the accelerometer sensor is connected. When the user tilts his hand forward, the car moves in the forward direction and when the user tilts his hand downwards, the car moves in the reverse direction. To steer the bot to the left and right, the user has to roll his hand accordingly.

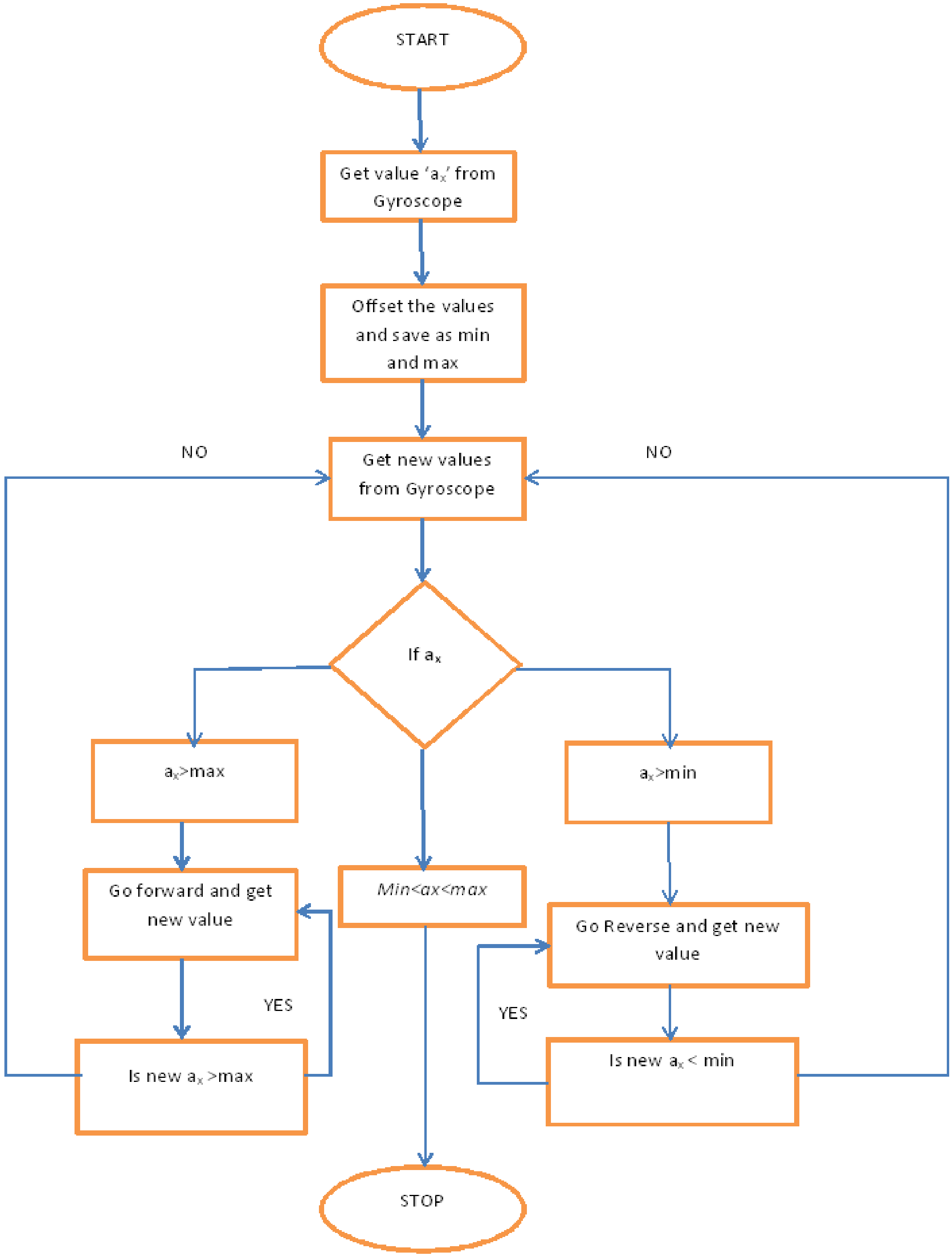
2) BLOCK DIAGRAM



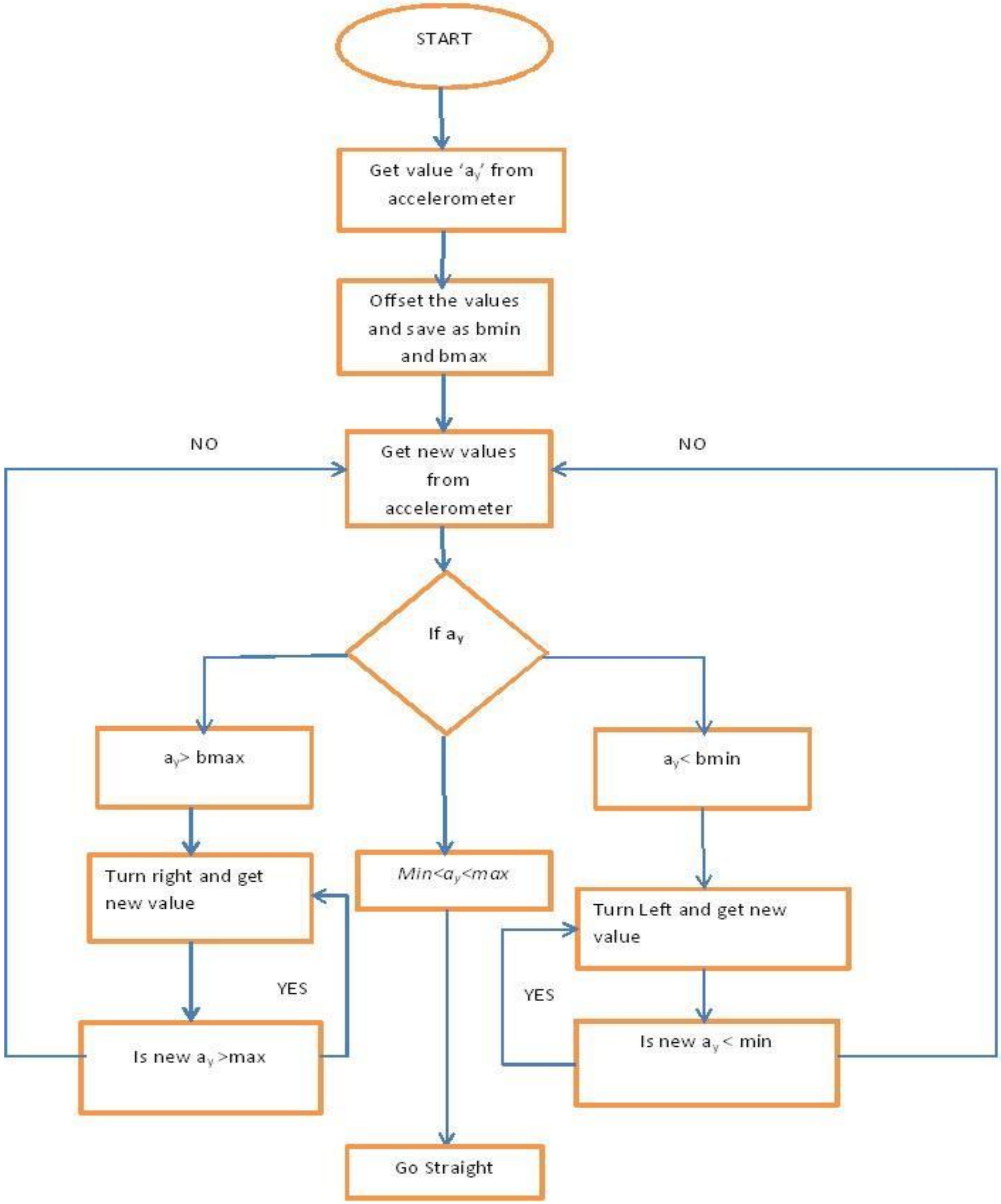
Block Diagram

The main components of our project are : MSP430G2553 is the heart of the project, accelerometer MPU6050 for sensing the hand gesture of the user, motor driver SN754410 for driving the motors and a servo motor to maneuver the car. When the user tilts his hand in any direction, the accelerometer senses the movement and transmits this movement to the processor in form of X and Y coordinates. The accelerometer continuously senses the hand gestures and communicates with the processor. The processor offsets the first value it receives every time it resets or is powered on and sets a threshold in order to reduce the sensitivity of the accelerometer. Depending on the X-axis accelerometer values, the processor drives the DC motors forward or reverse. Depending on the Y-axis values of the accelerometer, the processor steers the servo motor and the car moves.

3) **FLOW CHART**



Flow chart for controlling forward and reverse motion



Flow chart for controlling the direction

4) <u>DESCRIPTION OF COMPONENTS</u>

1) Accelerometer - MPU6050

The accelerometer used in our project was MPU6050.It is a 6-axis accelerometer and gyroscope module. The accelerometer uses I2C protocol to communicate with the microcontroller. The specifications of the accelerometer are:-

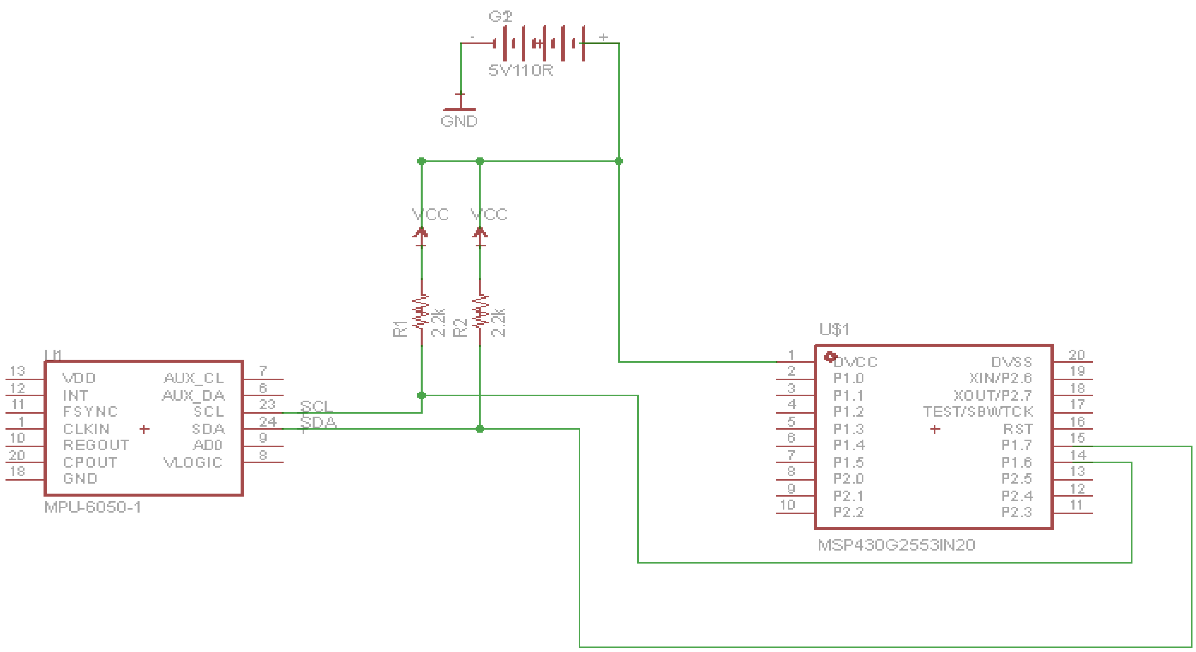
i) I2C Digital-output of 6 or 9-axis Motion Fusion data in rotation matrix, quaternion, or raw data format

ii) Input Voltage: 2.3 - 3.4V

iii) Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000 dps.

iv) Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g.

The pins P1.6 and P1.7 of the MSP430 are used to interface with CLK and SDA pins of the accelerometer sensor.



Hardware Interfacing of MPU6050 with MSP430

I2C Protocol :

-Master sends a start bit followed by 7 bit slave address.

-It is then followed by a single read or write bit.

-If there are any slaves in the line, the slave will respond with an acknowledge bit.

-Data transfer takes place.

-The slave sends another acknowledge bit.

-Master finally sends a stop bit.

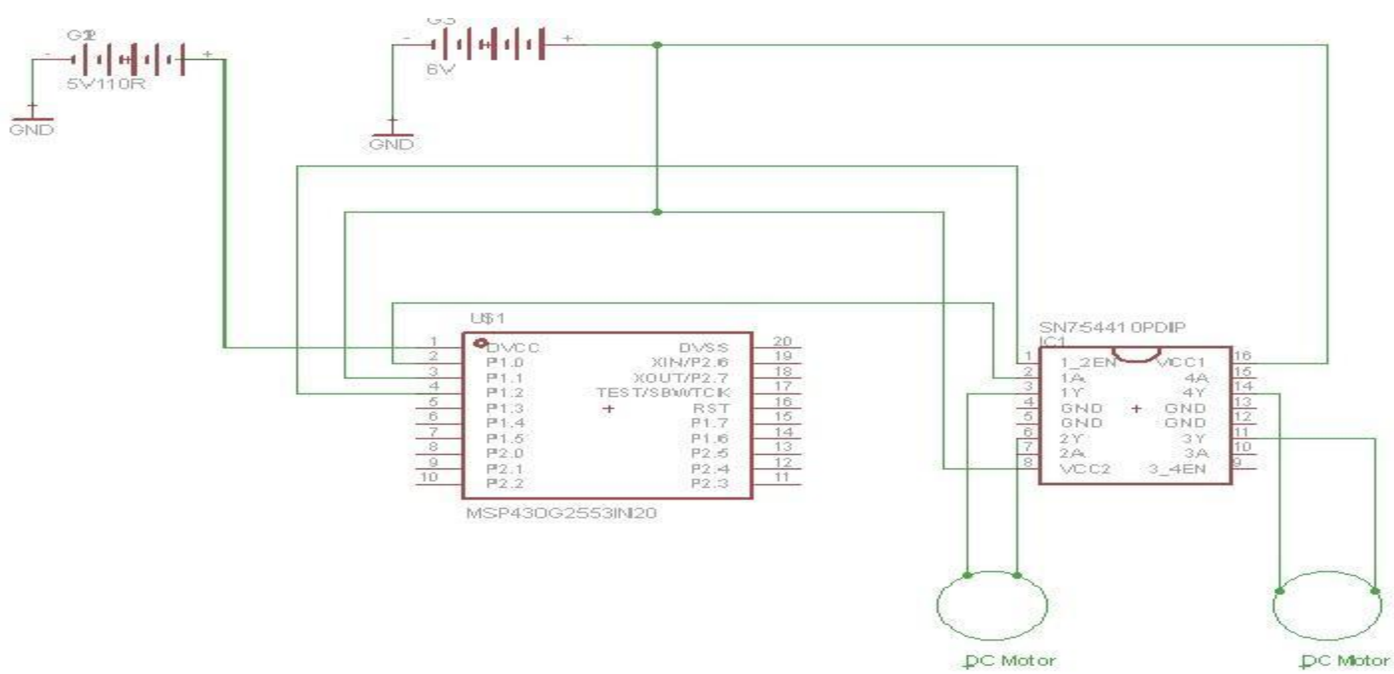
Initially a few writes are made into the MPU6050 registers to configure it.

The following procedures were done to initialize the accelerometer sensor.
i) Clock source was selected
ii) Full Scale sensitivity range was set.
iii) The sensor was woken up from sleep.
The following registers of the accelerometer are read to get the X and Y values respectively 0x3B,0x3C,0x3D,0x3E. The USCI_B0 data ISR is used to move received data from the I2C slave to the MSP430 memory. It is structured such that it can be used to receive any 2+ number of bytes by pre-loading RXByteCtr with the byte count.

We are using the pitch and the roll motion of the hand to maneuver the car. The raw values were very much fluctuating. It is impossible to keep our hands still at the horizon,the initial values change each and every time the program is reset. To prevent the car from moving right away, we came up with a logic where the first sensor value is taken and a threshold range is set,any hand movement within that particular range would not control the car. Each and every time the processor is reset a new threshold is automatically set.

2) H-bridge and DC motors :

We have used SN754410 motor driver IC for our project to drive the wheels of the car. A few important technical specifications of the IC are :
i)Full H and
ii) 1 A current output per driver
iii) Output Voltage Range – 4.5V – 36V



Hardware Interfacing of SN754410 with MSP430

We have supplied 6V to the IC from external battery. Pin 1(Enable) of SN754410 is connected to P1.2 of the MSP430. This pin is constantly supplied logic 1. Input pins 2 and 7 are input pins for the IC and they are connected to pins P1.0 and P1.1. Since both the motors run in the same direction at a particular time, we shorted the inputs of both the motors. Pins 3 and 6 of SN754410 are connected to the motors. To drive the motor in the forward direction, we gave pin P1.0 high output and to drive the car in the reverse direction, we gave pin P1.1 high output.

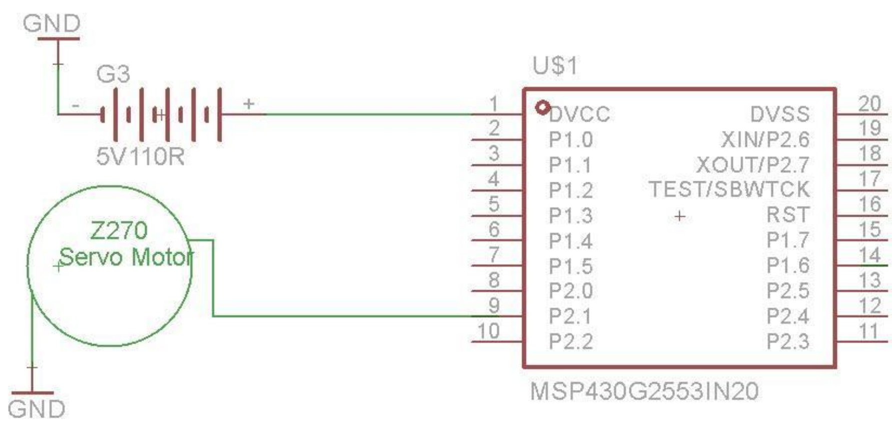
FUNCTION TABLE
(each driver)

INPUTS		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low-level
X = irrelevant
Z = high-impedance (off)

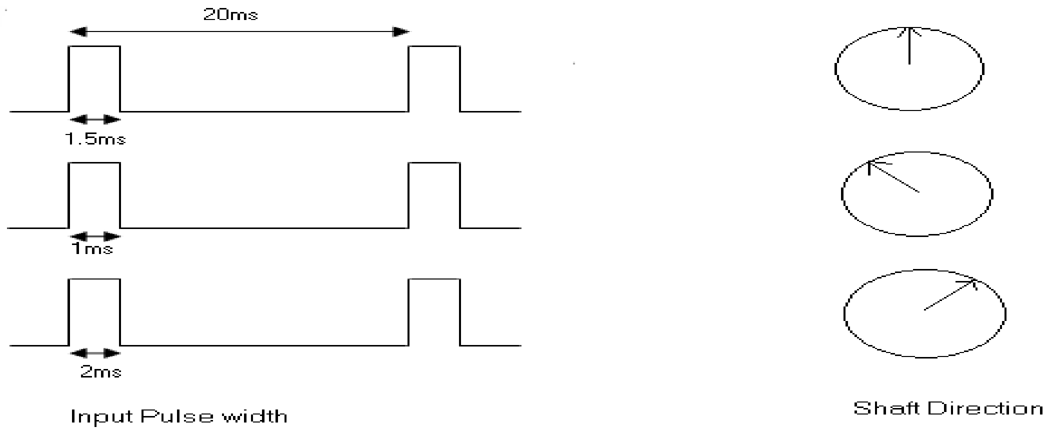
3) Servo Motor :

To control the direction of the car, we have used a servo motor Z270. The motor has 3 pins : VCC, GND and INPUT.



Hardware interfacing of servo Z270 with MSP430

The servo input is connected to pin P2.1 of the processor. Also, we have supplied the motor with 6V input from external battery. The input control to the servo motor is a pulse width modulated signal of 50Hz (pulse width of 20ms). By changing the width of the pulse we can control the angular position of the shaft of the motor. To hold the shaft in center position, a pulse of width 1.5ms is given to the motor. Changing the width of the pulse to 1.0 ms rotates the shaft to the left and a pulse of width 2.0 ms rotates the shaft and thus our car to the right.



5) <u>CONCLUSION</u>

We were successfully able to maneuver a toy car using hand gestures. From this project we learnt the working and programming of a microprocessor in real time. We also learnt the working of I2C protocol and were successful in implementing it . We also learnt interfacing a digital sensor to the microprocessor and its calibration. We also learnt working of the servo motor and its interfacing. We also learnt how to dynamically change the direction of DC motor and the Servo motors. This project gave us confidence to work on an entirely new platform.

Future Developments:

The accelerometer can be made to transmit values wirelessly to the processor. Also, cameras can be attached to the car and it can be used for data acquisition in hazardous environments, security purposes etc. Obstacle avoidance and pit avoidance systems can also be implemented.

APPENDIX

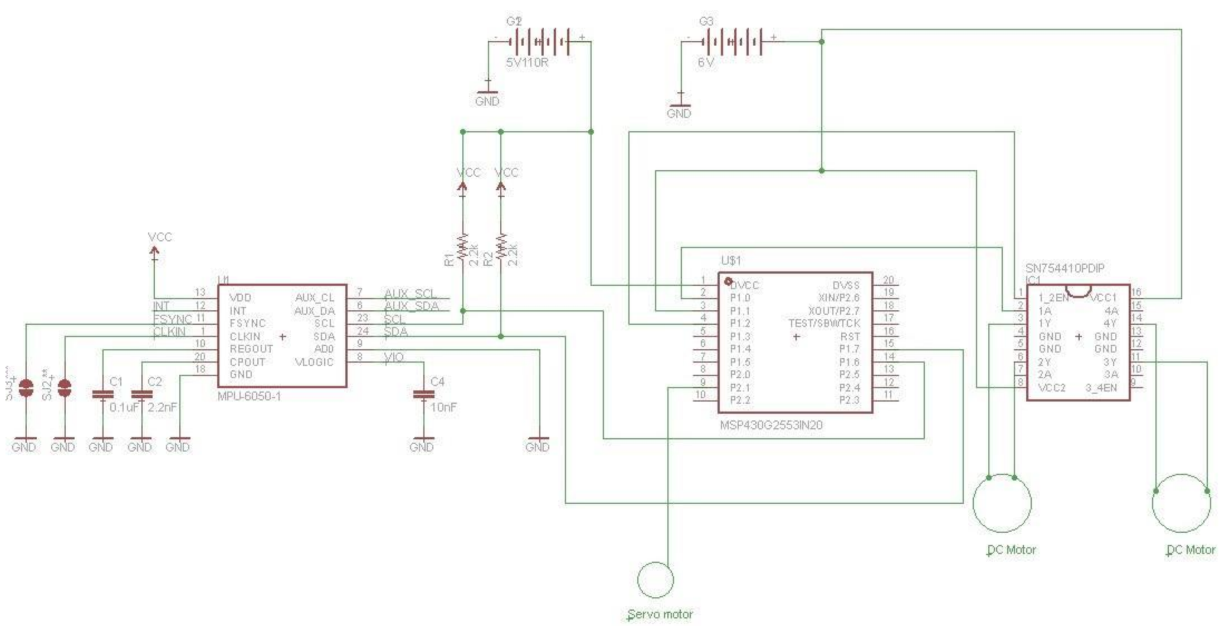
List of team members and their tasks

•Arvin Asokan – Gyroscope Interfacing and Calibration
•Urvi Tank- H-bridge Motor Interfacing
•Sai Kasyap – Servo Motor Interfacing
•Nikhil – Hardware and System Integration

Components List

MSP430G2553 LaunchPad	1 Qty
Accelerometer MPU6050	1 Qty
Motor Driver SN754410	1 Qty
DC Motors (200 RPM)	2 Qty
Servo Motor Z270	1 Qty
Battery 6V	2 Qty
Resistors : 2.2K (pull up for I2C),	2 Qty
Breadboard	
General Purpose board	1 Qty
Push Button	1 Qty
Glove	
Connecting Wires	

SCHEMATIC DIAGRAM



References

1. http://forum.43oh.com/tags/forums/I2C/
2. http://www.robotplatform.com/howto/L293/motor_driver_1.html
3. MSP430 Microcontroller Basics by John Davies (Author)
4. http://en.wikipedia.org/wiki/TI_MSP430
5. http://www.ti.com/tool/ccstudio-msp430