CSE316: Microprocessors, Microcontrollers, and Embedded Systems

Project Name: Connect Four

Team Members:

Student ID	Name	Mobile No.
1305031	Md. Rukshar Alam	01950023164
1305057	Md. Saqib Hasan	01670259917

Submission Date: 9/6/2017

Youtube Link of Video:

https://www.youtube.com/watch?v=6vwa2kxnJWQ&feature=youtu.be

Features Implemented:

1. We were supposed to implement two options to play, player 1 vs player 2 or human vs computer. But we have implemented only player 1 vs player 2.
2. The control is implemented using a personally designed keyboard made using gyro sensor(GY-521) and breadboard.

Features Not Implemented:

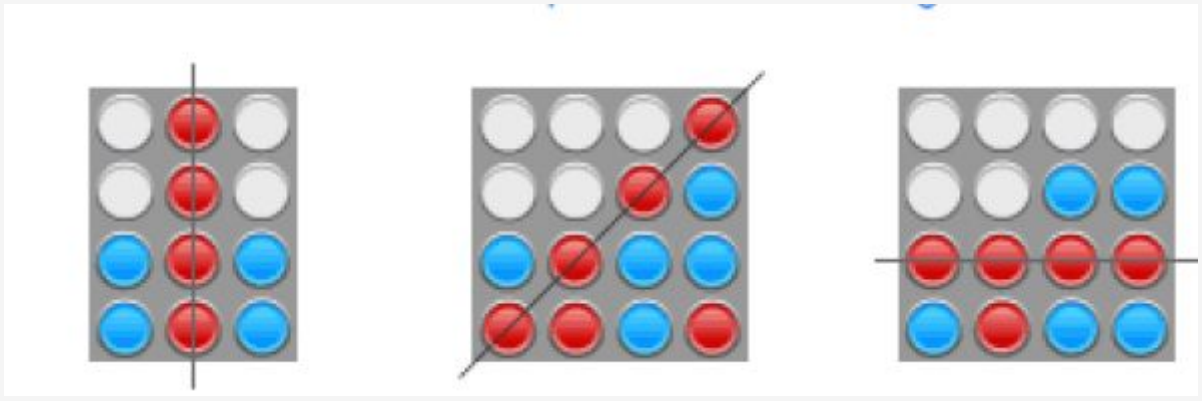
We could not implement human vs computer mode due to lack of time.

Basic description of the Project:

Connect Four is a popular strategy game played by people of all ages. It is usually played using a 6 X 7 hole board with two sets of coloured chips.

Rules of the game:

1. The game is played between two players
2. Each player drops a chip through one of the 7 openings at the top.
3. The one who can match four chips of their color in a row wins the round.
4. The match can occur vertically,horizontally or diagonally.



Basic Working Principle (Flowchart & Circuit Diagram):

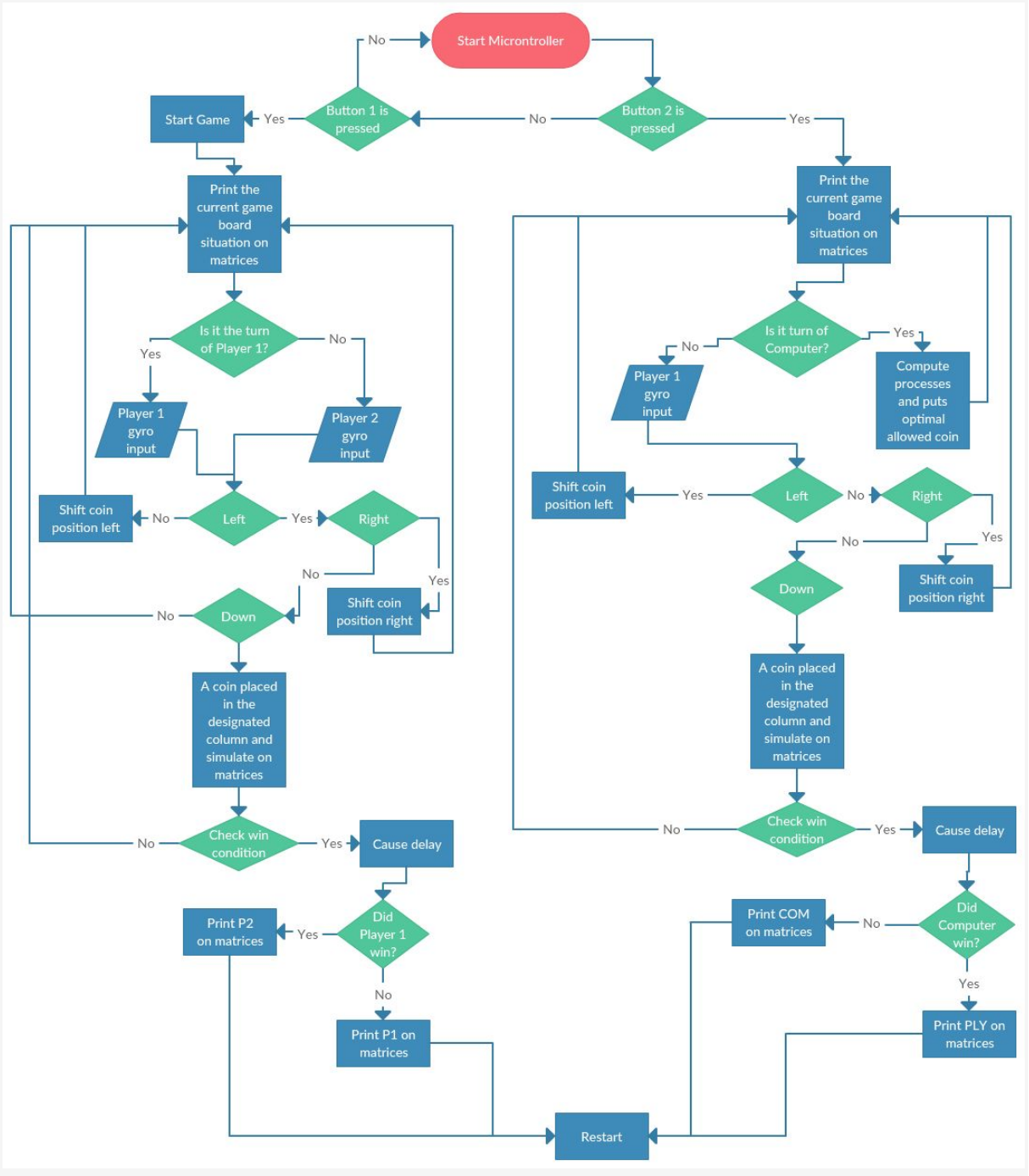


Figure: Flow Chart

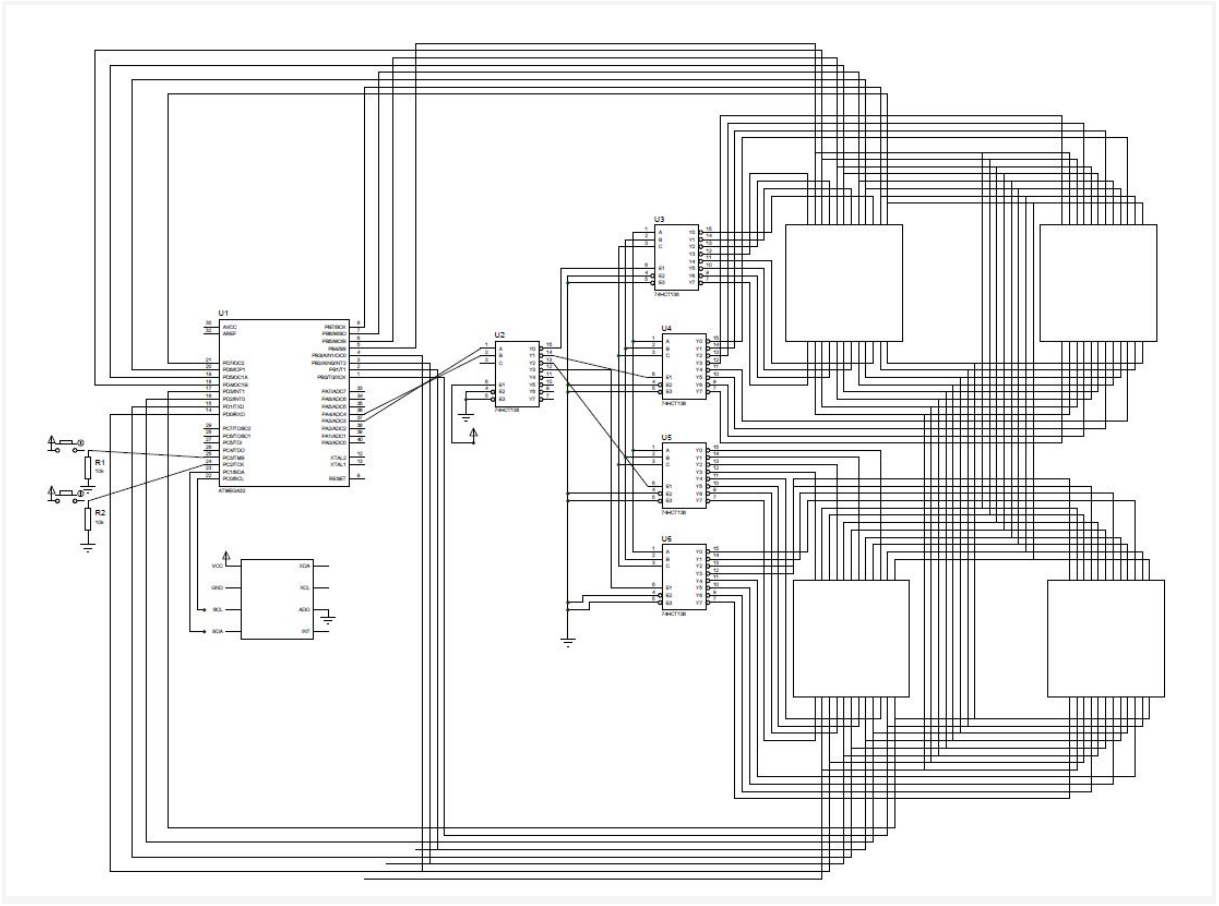


Figure: Circuit Diagram

Interfacing different sensors and Using the libraries:

We have used GY-521 in our project.

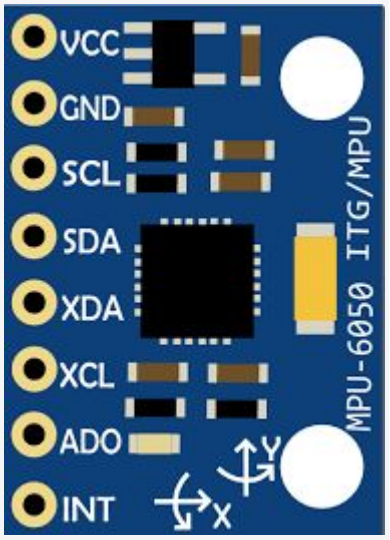


Figure: GY-521

The MPU-60X0 has an embedded 3-axis MEMS gyroscope, 3-axis MEMS accelerometer and Digital Motion Processor™ (DMP) hardware accelerator engine. The MPU-60X0 features three 16-bit analog-todigital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs.

Pin connection:
1.VCC to 5V,
2.GND to 0V,
3.SCL to Atmega32's SCL Pin
4.SDA to Atmega32's SDA Pin
Libraries:

MPU6050_res_define.h: In MPU6050_res_define.h, some constant parameters are defined as for my convenience.

I2C_Master_H_file.h: This file contains some important functions for communicating and reading values from different registers of MPU6050. I2C_Init() method initializes the communication. I2C_Start_Wait() method takes slave device write address as argument and returns different unsigned integers to indicate ack or nack received or failure. I2C_Repeated_Start() method generates REPEATED START condition for read operation. It takes slave device read address as argument and returns status of event. I2C_Write() writes data/address on bus. I2C_Read_Ack() method reads data available on SDA line and returns its acknowledgement to slave device about successful data read and also tells slave to transmit another data. I2C_Read_Nack() method reads last needed data byte available on SDA line but does not return acknowledgment of it. It is used to indicate slave that master does not want next data and want to stop communication. I2C_Stop() method initiates stop condition.

mpu6050_twi.h: It has been written especially for MPU6050 device. Gyro_Init() method is the gyro initialization function. It uses previously defined I2C_Start_Wait(),I2C_Write() and I2C_Stop() methods. MPU_Start_Loc() method initializes data transfer between gyro and ATmega32. It uses previously mentioned I2C_Start_Wait() ,I2C_Write() and I2C_Repeated_Start() method. Lastly comes the method Read_RawValue(). This method reads raw data from MPU6050.It reads all data from X axis acceleration to Z axis G acceleration serially. All are in 2's complement form in the registers. This method uses previously mentioned MPU_Start_Loc(), I2C_Read_Ack() , I2C_Read_Nack() and I2C_Stop() method . This function is continuously called from the main code to get the accelerations caused due to tilt in the X, Y and Z axis. 6 registers each of 8 bit in size are used for these values. ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_XOUT_L, ACCEL_ZOUT_H, ACCEL_ZOUT_L are these 6 registers.

Problems and challenges we faced and how we overcame them:

1.Whether individual parts and codes work properly. done by unit testing.
2.Gyro does not work most of the time. So we had to take readings or recorded the times it did work.
3.LED does not light up properly. Tested delay using trial and error to solve this
4. Coding for 32 LED using decoder. At first not all LED lights were working properly because there were wire entanglements and misplaced wires. Through changing delays and testing, and structural design the debugging was made easier and so we were able to overcome this challenge.