Problem Statement

The main aim of this project is to design a bot which can balance itself on two wheels. This project was the Eyantra (IITB) problem statement assigned to my team. We were selected till hardware round and unfortunately, we could not make the bot completely working. But the bot managed itself to make stable. Following report illustrates the entire idea.

The structure comprises of an axle. On its two sides wheels are connected. The whole chassis is then placed on this axle. By common understanding the chassis will topple. So, in order to make chassis stable we need to move the wheels. The idea could be more simply visualised as having a stick upright in our palm and we move palm to and fro to balance it. But the question is how fast it should be? The answer is to use a controller algorithm. In this project LQR controller algorithm was used.

Components Used

Hardware Components:

- Arduino Mega
- Breadboard
- XBee
- Toggle Switch
- RGB LED
- Electromagnetic Module
- Geared DC Encoder Motors & Motor Driver Module (L298)
- IRF540N MOSFET
- GY-87/HW-290 Sensor Module

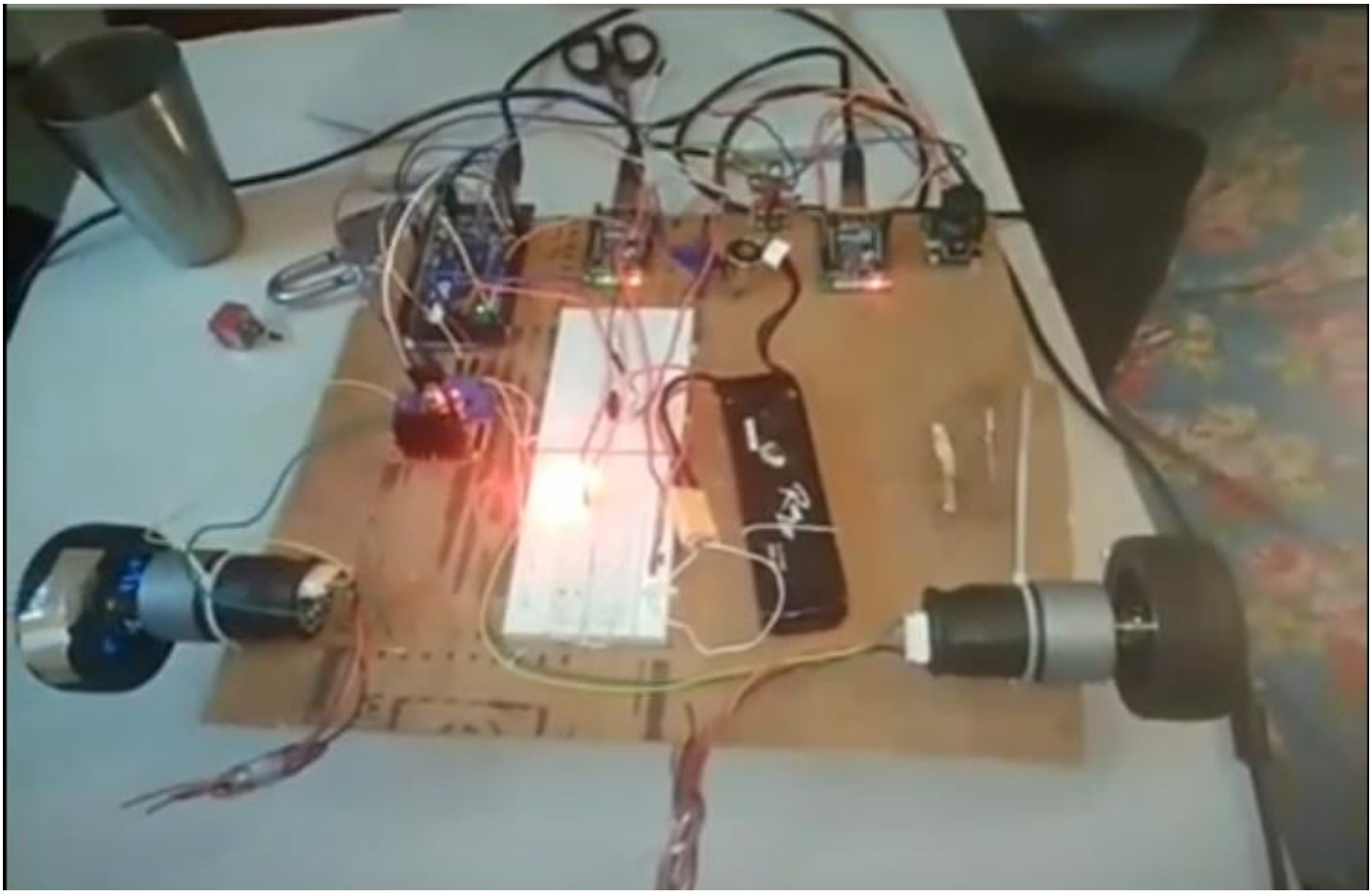
Software Components:

- MATLAB
- Arduino IDE

Making Wireless Communication

The first step in the hardware designing was to make wireless communication. We need to make a remote. So, the main idea was to configure XBee transceiver module. That is to decide which one will be transmitter and which one will be receiver. After that it is necessary to send data of receiver and process it. To understand the frame received we need to look in particular bits of the frame in order to decide which operation needs to be done. As for example, consider user made pin 1 of transmitter high. This information is received by receiver, but it gets a frame. To understand it Arduino code tests each bit and then performs respective functions.

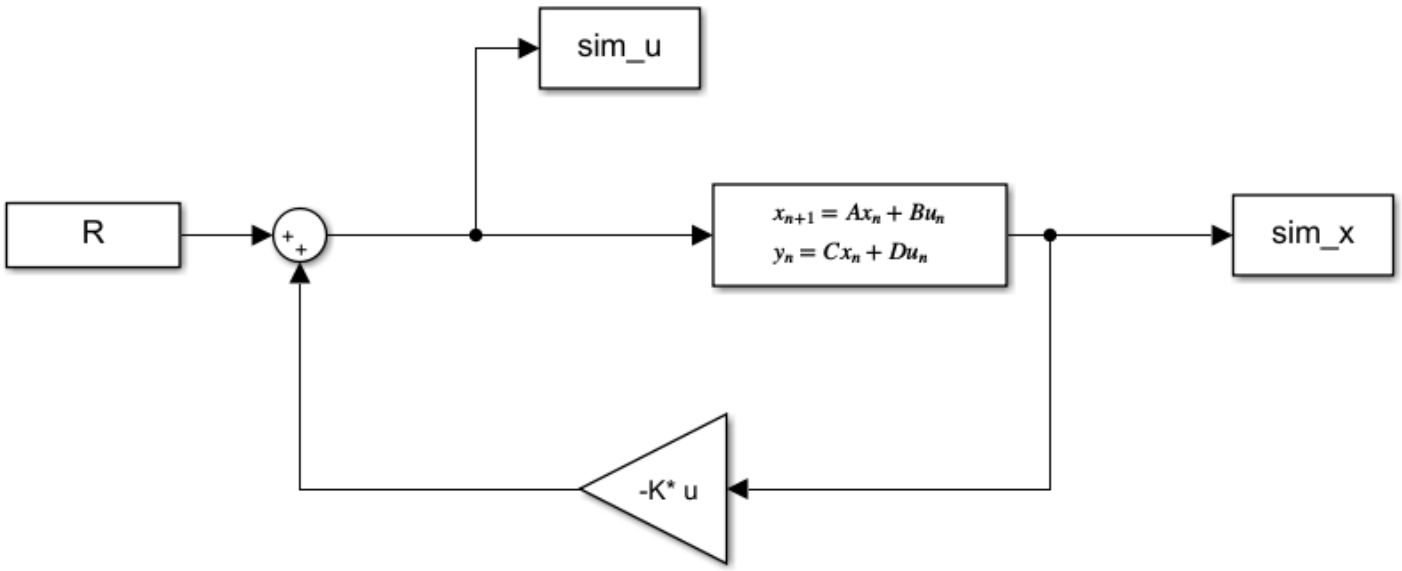
There are functions like glow LED, run the motor forward- backward, control the electromagnet etc.



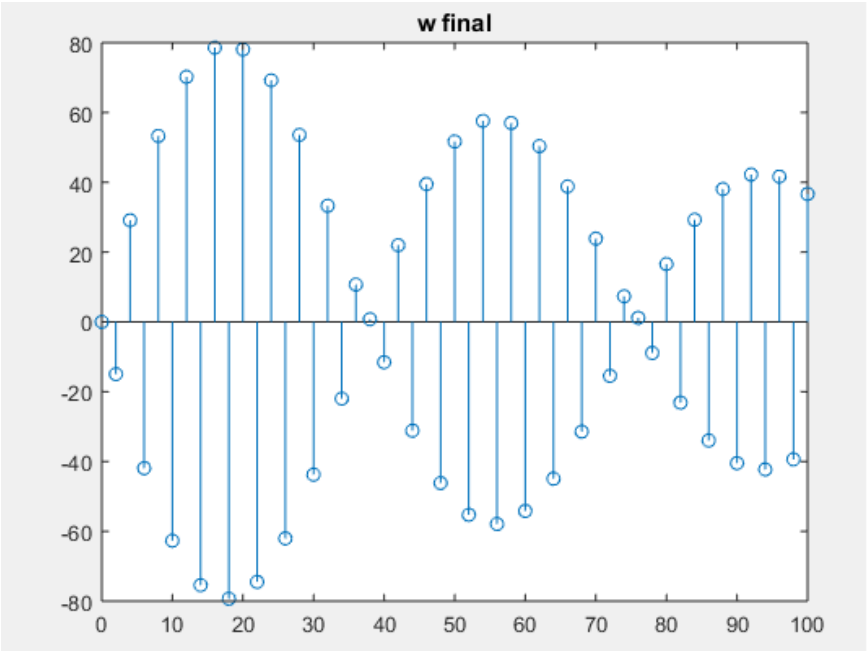
LQR Designing

The control system algorithm required knowledge of previous state condition of device, based on that it calculated required motor speed and make the desired angular rotation. So, the variables were position, current speed and angle of tilt. The angle of tilt was measured through Gyroscope. Simulation was performed through MATLAB.

Following is the control system block diagram:

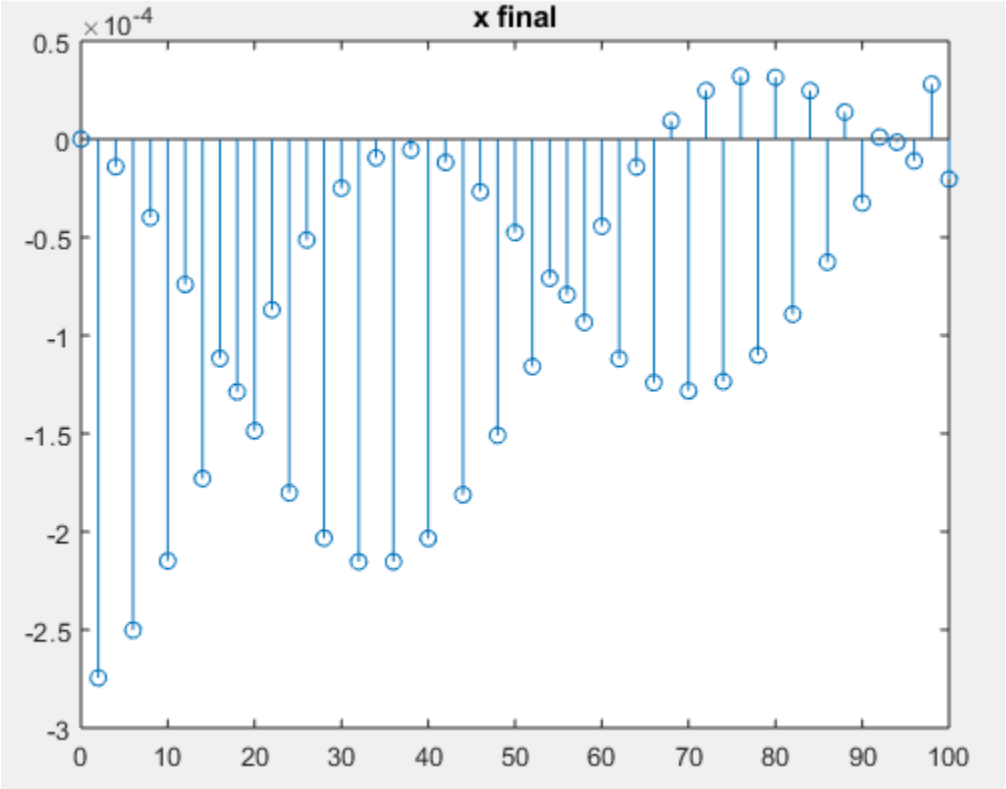


Following is the simulation:



It is evident here that the motor is rotating to and fro.

The bot is oscillating about the origin:



But the question is how to implement it in Arduino? The solution can be given by observing the state equation,

$$x_{n+1} = Ax_n + Bu_n$$
$$y_n = Cx_n + Du_n$$

The state equation constants could be found from the Simulink. Then by using above equation we could determine exactly what numbers we need to multiply in order to get the next state value, that is how much should be the rotation speed that is W_{n+1}.

Gyroscope Reading Filtering

Now due to response time differences between accelerometer and gyroscope we need to use filter. As if response time is slow then we can say the value changes less frequently. On the contrary if it has fast response time then we can say the output signal is having high frequency component. Now accelerometer is slow device hence low frequency components should come whereas gyroscope is fast therefore high frequency component should come.

Thus, we could say that we need high-pass filter for gyroscope and low pass filter for accelerometer.

How to implement on Arduino? The implementation can be done by using following equations and then use the updated value.

Low pass Filter:

$$y[n] = (1-alpha).x[n] + alpha.y[n-1]$$

x[n] is the pitch/roll/yaw that you get from the accelerometer
y[n] is the filtered final pitch/roll/yaw

High Pass Filter:

$$y[n] = (1-alpha).y[n-1] + (1-alpha)(x[n]-x[n-1])$$

x[n] is the pitch/roll/yaw that you get from the gyroscope
y[n] is the filtered final pitch/roll/yaw which you must feed into the next phase of your program

n is time instant.