PROJECT REPORT

Group no. 140
Varunesh Goyal-140070006 (Group Leader)
Devesh Khilwani-14D070045
Saurabh Chavan-14D070036
Saurabh Pinjani-140070056

Table Of Contents-

SKETCH BOT

Introduction:

The project idea is designed around a program which is capable of processing any given input image, and reproducing the same physically. It can make working of automated systems in numerous industries a lot easier and provides solutions to various problems faced by them. The current project idea is a form of the real world robotic systems. It can not only recreate the input image given to it, but also it can take real-time input from the user in which the user himself draws the input and send it to the bot.

One of the useful features of this project is the two way communication between the bot and the UI through XBee. It makes a kind of feedback system having many applications.

Problem Statement:

The aim of the project is to design an automated bot (using Firebird V) which would draw an image taken as input from the user.

We aspired to process the image given as input by user and extract a path from it using Image Processing techniques.The path would be relayed wirelessly to the bot.

The bot would then traverse the required path and replicate the digital image boundaries in physical realm by a marker attached to it.

Requirements:

A)Hardware Requirements

1.FireBird V	:Bot used to draw the image.
2.Servo motor	:Mounted on the bot
3.Marker	:To draw the image
4.XBee	:For wireless communication between UI and bot

B)Software Requirements

1.OpenCV	:Library for processing input images
2.AVR Studio 6.0	:For writing Atmel Code
3.AVR Bootloader	:loading atmel code in the bot
4.Code::Blocks 13.12	:IDE for writing the C++ code
5.Eclipse	:IDE for writing the Java code (UI)
6.JDK 7	:Compiler for Java

Implementation and division of work:

We realised that for doing this project we should have a sound knowledge of opencv libraries including vector class, file handling, java, and embedded C functions. So we learnt more about these topics from different sources and references that were available to us. We made a rough sketch of the various ideas and concepts that would be used in the project on paper. After a lot of discussion we summarised our entire program into four parts:

1. Processing of Image to give required points.
2. Preparing UI to give input image.
3. Working on the bot related functions.
4. XBee Integration.
5. UI and C++ program integration.

We distributed the workload amongst ourselves as follows:
1. Saurabh P would prepare the user Interface in java. In addition he would also contribute in writing some of c++ code. eg, merge path function and SRS.
2. Varunesh would look after the processing of image in opencv to detect the borders and generate a vector of points (the path to be followed) and send it one by one to the bot through XBee.
3. Saurabh C would prepare the motion control code for bot and also for receiving data through XBee. Along with Devesh, he would also write the code for servo motor control.

4. Devesh would work on integrating the UI with c++ code, Servo functions, make the presentation and also contribute to some of image functions and XBee.

In this entire project we all were together and we had support of each other all the time. Final debugging was done by all of us together. Our TA Samson Khess also helped us when we needed.

Challenges:

1. **One of the major challenges we faced was using XBee for wireless data transmission and reception with bot.**
   The Problem : - Linking data from cpp with Atmel code was difficult.
   Solution : - We later sorted out it by using windows.h library to write data (one byte at a time) onto the COM port to which XBee was connected. The Atmel code contained instructions for the bot corresponding to the data received on the port.

2. **Secondly, we had problems connecting User Interface with the image processing code in cpp.**
   The Problem : - Major issues were encountered while sending data from interface to cpp code. Data contains address, resolution and input mode. It took time to realise that even on sending these data separately it would be received by cpp code as a single string, moreover data was sent only upto first space character.
   Solution : - We replaced spaces by '!' and separated different data by ',' in JAVA based user interface as '!' and ',' are uncommon in addresses. The required data is then extracted

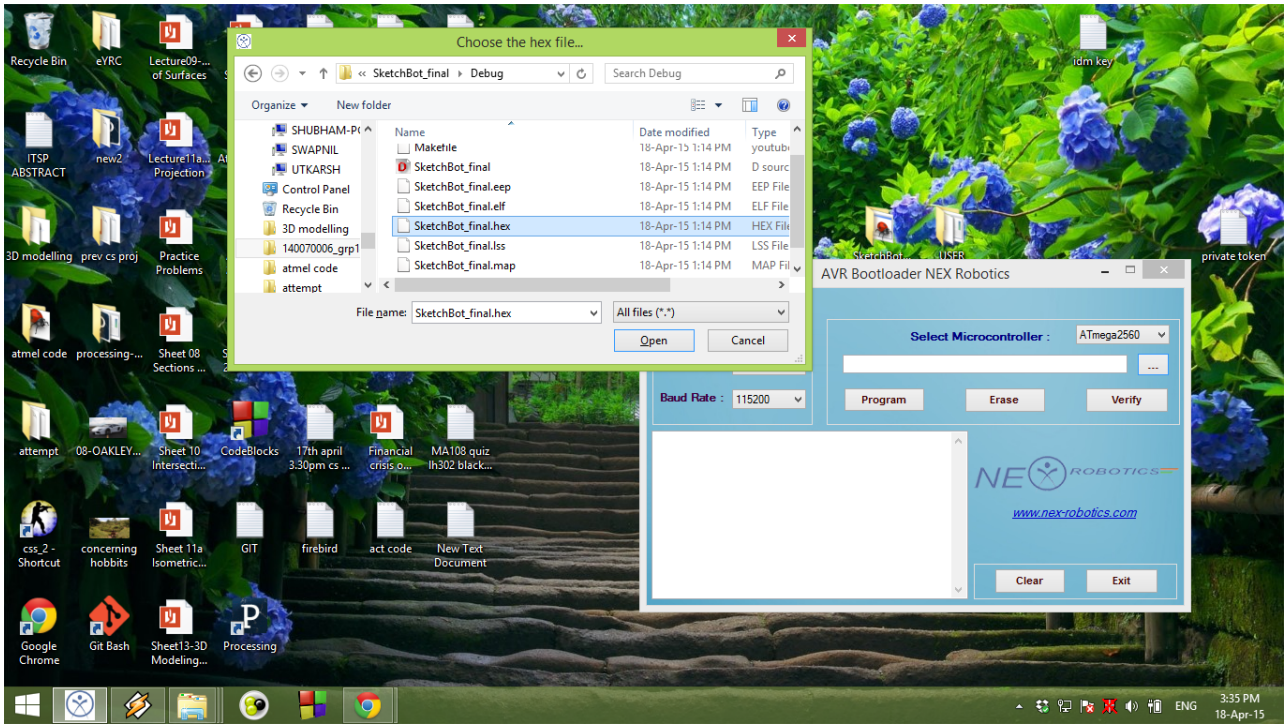
from the single string using suitable algorithm.

3. **Extraction of optimal points from boundary for the bot to travel has been a challenge throughout.**
   The Problem : - On including all the boundary points some points would be closer than the bot could least move. Moreover, greater number of points meant greater starts and stops, leading to greater slipping and thus, greater inaccuracies.
   Solution : - We had to optimise number of points to maximize accuracy. We did this by including only those points which are far enough to be travelled by bot. We also delete unnecessary points which lie on same line by taking only the end points of the line.

User Instructions:

To use this project the user need to follow the steps given below:

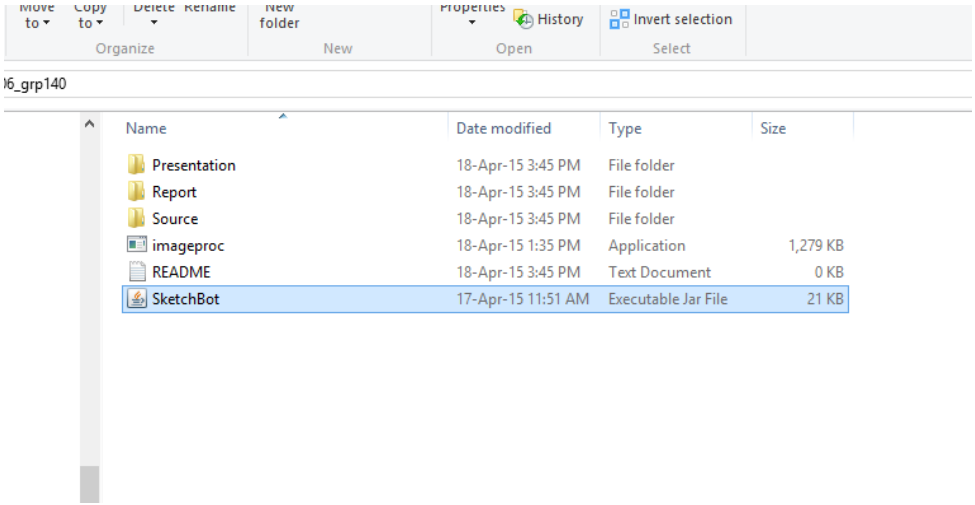
1.Firstly load the file SketchBot_final.hex into AVR Boot loader.The file can be found in the folder “Source\SketchBot_final (Atmel)\SketchBot_final\Debug”. Set the bot firebird V in program mode,and program it.



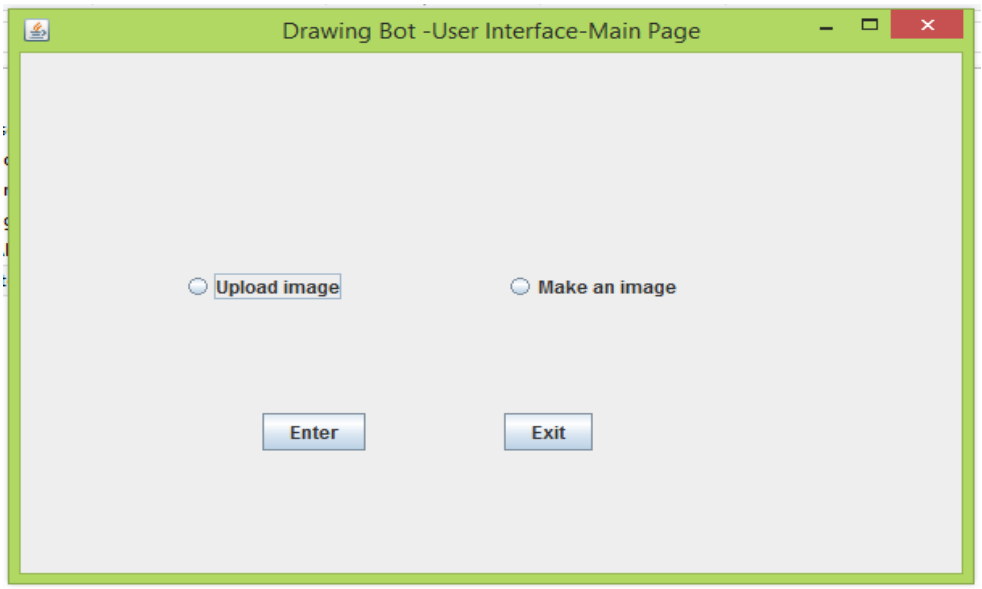
2.Next connect a XBee Module in port COM1 of your PC.
If not go to Device Manager. Then under Serial Ports right-click on the device, go to properties and under Advanced options change the COM port to COM1.
Assuming the XBee is configured(you can configure it using XCTU software), press the reset button of the bot.

3.Ensure that the pen is properly connected to the bot i.e. touches exactly at the center of the axis of the wheels in penDown position.

4.Run the java executable file SketchBot.jar.

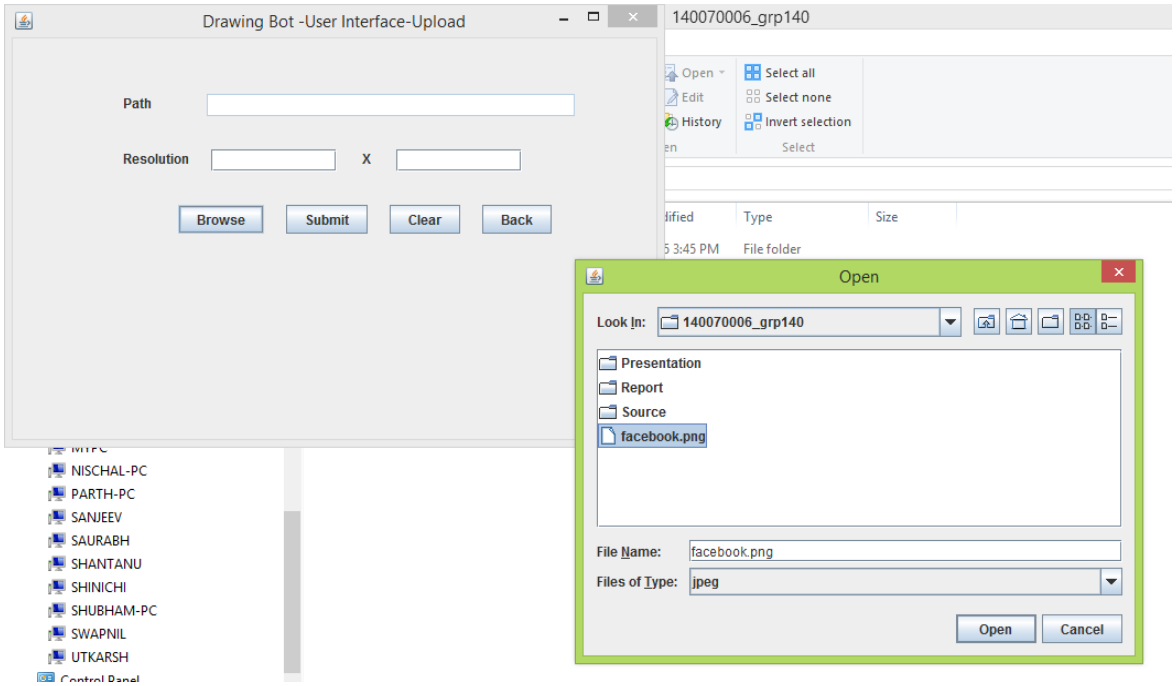


5.Select one of the two options in the UI..



If you choose Upload an image:

- Click on browse and select the image path. You need to ensure that the file named imageproc.exe(which can be found in the main directory of 140070006_grp140)is present in the same folder as that of the image i.e. copy the file imageproc.exe into the folder containing the image.



- Give the appropriate resolution and click submit.
- The original image as well as the image consisting of only the detected borders which will be drawn is shown.
- If the user doesn't want some part of the border to be drawn, he can erase that part by dragging the mouse over the region while keeping the right mouse button pressed.
- On pressing Enter, the bot will draw the given image,and a message "successfully drawn" will be displayed after that.

If you choose make an image:

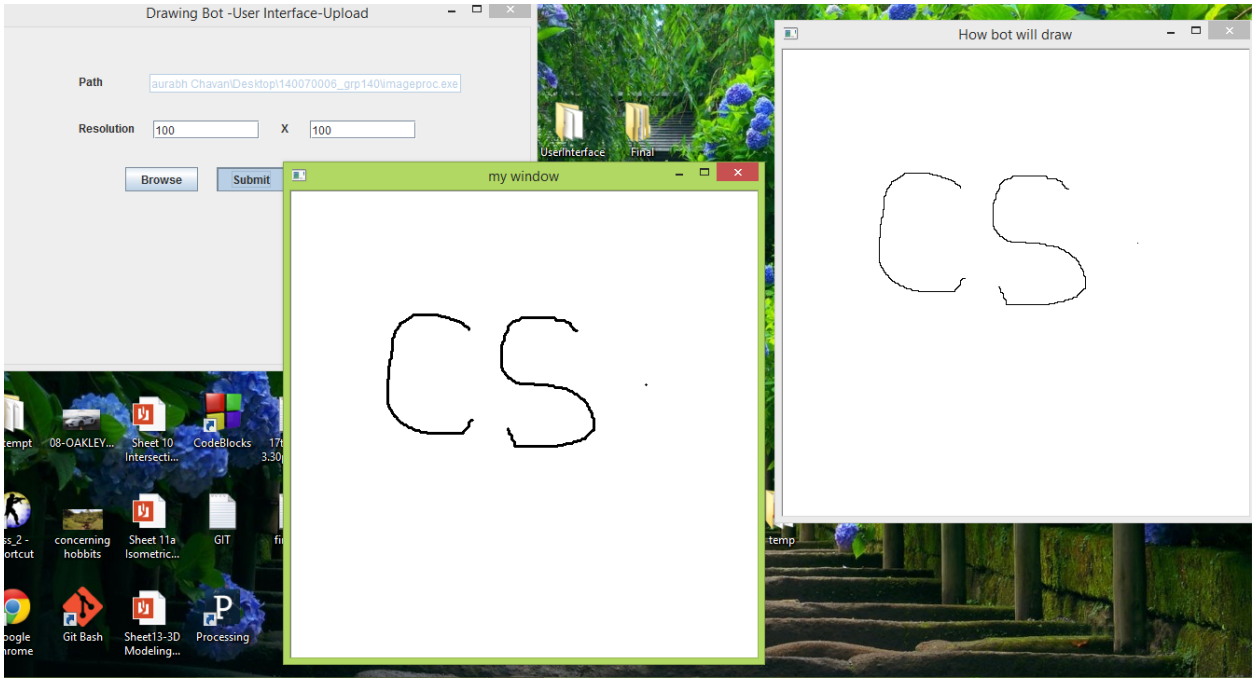
- Click on browse and select the file imageproc.exe from the main directory of SketchBot.
- Draw the image in the window displayed and press enter.
- The bot will draw the given image,and a message "successfully drawn" will be displayed after that.

6.Ensure that you press the reset button of the bot every time you give a new input to the interface.

Testing Strategy and data:

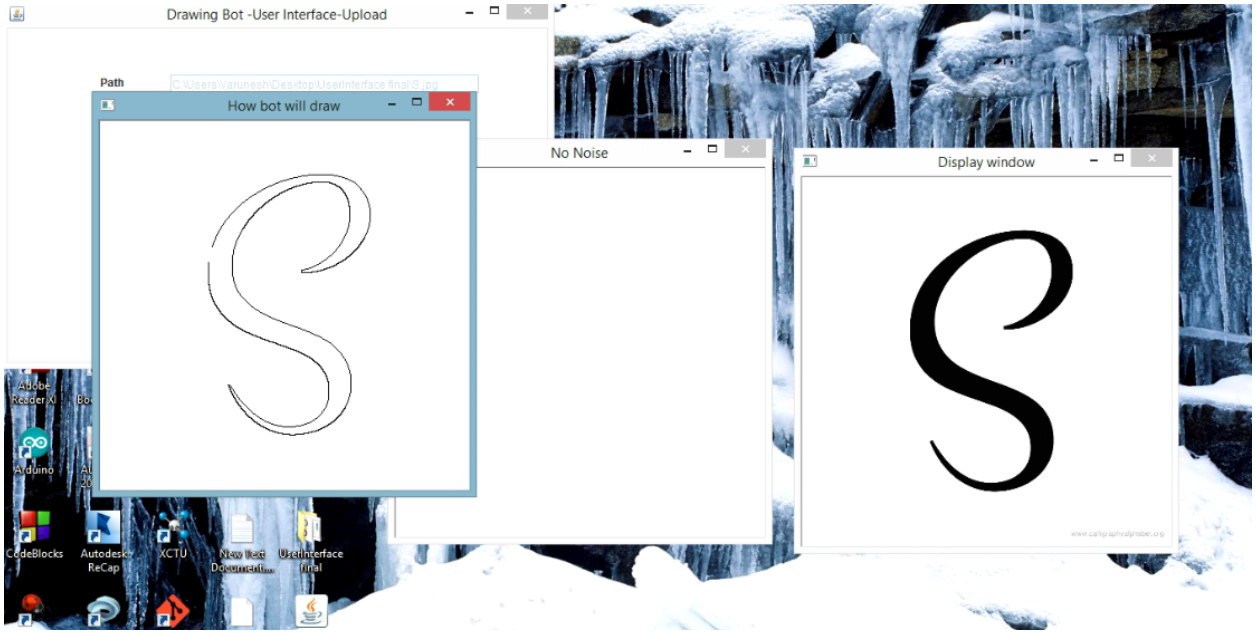
We tested the bot for a wide range of input images and drawn images.Our bot drew them to good enough accuracy.Some of the images our bot drew are as follows:

When we drew image of CS,

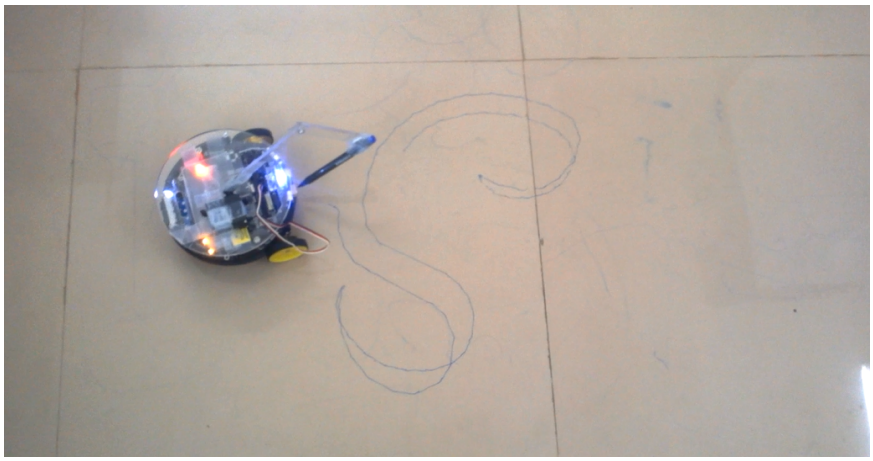


The bot drew this :

When we uploaded an image as follows,



The bot drew in this way.



Discussion of System:

A) What worked as per plan?

1.Our bot could successfully drew all the input images given to it besides basic shapes,of course with slight error.

2.Zigbee Communication:The ZigBee communication was quite efficient as the response time of the bot for receiving and transmitting signal was very less.We were successfully able to communicate with the required bot by using Zigbee and able to take appropriate action at real time.

3.Servo Function: The servo motor also functioned well.It would do pen up whenever there was a discontinuity in the path and pen down whenever required.

B) What we added more than discussed in SRS?

1.We added the feature to draw the image at real time through user interface, though it was mentioned as a side project in the SRS.The bot would then draw this image.

2.We introduced a function named “Erase” using which the user was able to remove unwanted path of the image just by right clicking at real time.

3.We added support for images drawn with intersecting lines.

4.The bot is also able to traverse curved paths approximately (well actually pretty accurately)

5.We managed to create a well working User Interface in java which made giving input to the bot simpler.

Future Work:

- The bot can be used in fabrication industry. The required side view can be be given to the bot as input which would then cut the material in required shape.To fulfil this purpose, the marker can be replaced by a laser or any cutting instrument.
- The bot can be input a path from a map. It could then be used to travel over the path and fetch relevant data to be used for statistical purposes and resource mapping.
- The bot can be used to tread over paths those are otherwise difficult to be travelled by humans.

Conclusion:

Our bot is a general path follower type robotic module. It can be used in varied fields which need this path following feature. The uses can be as diverse as fabrication, goods delivery systems, exploration.

References:

1.OpenCv and Code::Blocks 13.12 installation guide
"http://kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw/"

2.OpenCV tutorials
"http://docs.opencv.org/doc/tutorials/tutorials.html"

3."Related-Videos"folder in main directory of project for video tutorials and working bot videos.

4.FireBird tutorial
"http://www.e-yantra.org/index.php/eyantra/tutorial"