# Tomato Harvestor

## CS684 Course Project Report

Amandeep Chugh (123050017)
Ashish Kumar Agrawal (123050023)
Jayasree Kaveti (123050050)
Lorin Ahmed (123050029)

November 18, 2012

## 1. Introduction

In a typical green-house environment, many simultaneous and series of actions should keep going on to keep the plants in it healthy. These actions include, seeding, watering, nutrients feeding, harvesting, weeds removal etc. Therefore its better to automate all these processes in the green house. There is possibilty of having separate bots for performing separate action.

Our bot "Tomato Harvestor" is designed specifically for harvesting of tomatoes in a green house. It traverses the specified aisles as per the user instructions and searches for any ripened tomato in the troughs. Which on detecting a tomato, cuts it with its cutter, collects it and comes back to its intial postion for deposting it in the tomato repository.

## 2. Problem Statement

One of the major tasks involved in maintaning Green houses is harvesting. Our project automates this task specific to Tomato plants. The programmed bot goes to any trough in the green house specified by the owner through the User Interface, processes that particular trough and comes back. Processing the trough involves three tasks majorily.

1. Moving through the trough.
2. Detecting Ripened Tomatoes.
3. Cutting nad Collecting them.

A Graphical User Interface on a smart phone which helps user specify the trough number which needs to be processed. This interface should run a backhand program which connects and disconnects with the Bluetooth module of Firebird Robot. The smart phone is attached to the bot. When the bot reaches the trough specified, the task of image processing starts on phone. When it detects ripened tomato, it signals the bot through bluetooth the actions it should take to cut and collect the fruit.

# 3. Requirements

## 3.1 Funtional Requirements

The Tomato Harvestor use the camera of Andriod Phone to detect ripened tomatoes in the trough by running a program of Image processing coded in opencv in java on it. When a tomato is detected, the algorithm calculates the distance to travel for the bot and the amount of Pulley's vetical movement to bring the tomato in the center of the screen, after which the cutter will get on and te tomato will be cut.

The specific trough to scan for the tomato will be known by the signals received from the remote Android based UI using XBEE (Based on zigbee protocol, IEEE 802.15.4) which gives information about the no. of trough.

## 3.2 Non-Functional Requirements

1. Cutting and collecting the tomato without damaging it.
2. Robot Consuming less energy during the going to the trough, cutting the fruit and coming back.
3. Tomato detection should work in bad lighting conditions as well.

## 3.3 Hardware Requirements

1. Firebird V Robot.
2. Android device.
3. Zigbee Module.
4. Bluetooth Module.
5. Two Servo Motors.
6. Whiteline Sensors.

# 4. System Implementation

We have 2 major components viz. Android phone and fire-bird which communicate with each other via blue-tooth protocol. Fire-bird is the master controller which takes input from zig-bee and gives/relinquishes control to/from android phone.

### 4.1 Android code Description

Android phone takes charge of initiating the initial handshake with bluetooth module present on the bot[2].
After Camera is turned on, the phone camera constantly monitors the view for any pink object ( pink tomato simulated) and sends appropriate locomotion, pulley and cutter on/off instructions to the bot over bluetooth.

**Tomato detection algorithm**

The image processing algorithm first detects tomato and then move the bot and the pulley appropriately to align the tomato in front of the cutter in the bot. This alignment works by dividing the captured frame into 9 boxes. The algorithm tries to align the bot with the tomato such that it comes inside the center box of the frame.
In the first phase, Frame is considered to have 3 vertical strips, and the robot tries to put the tomato in the central strips. After which Frame is thought to have 3 horizontal strips , pulley is moved up or down to bring the tomato inside the central strip. In the end of these two phases, the tomato comes actually inside the center box and thereby comes in the position to be cut by the robot.

### 4.2 Firebird code Description

Fire-bird code has two modes, following the black line to respective trough and taking instructions from android phone camera for locating the camera.
The remote monitoring device ( over Zig-bee) communicates the trough number and the speed of the bot. After receiving the signal, the bot gets itself to mode 1 and reaches the respective trough. After reaching, it gives the control to phone camera and waits for it to locate and cut the tomato.
Finally it retraces back to the origin from the end of the trough.

**Setting up OpenCV4Android**

Basic Setup and neccessary installables[1] required before OpenCV library can be used for feature detection ( pink sphere in our case) can be found at:
''http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/''

## 5. System Description

The system consists of one Andriod phone, a Firebird V Robot and a remote Zigbee Controller. All the Image Processing and Instructions for the robot while traversing

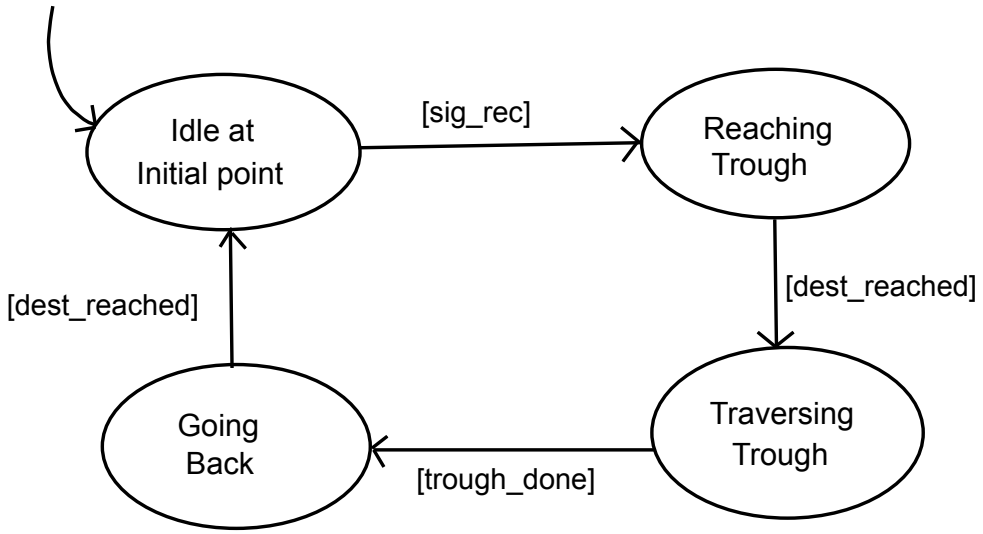


Figure 1: Big Picture of the System

the trough are being calculated on the Android Phone. Remote Zigbee Controller sets all the initial parameters needed to get the robot started going towards the specified trough, harvest tomatoes there and come back to the initial position. And the Firebird robot will be actually moving to the trough as per the remote user command, performing its task there and coming back. The camera of Android phone itself is used for video capturing.

The Remote Android Controller will be broadcasting the instruction packets using XBEE, where robot will recieve message as packets and will get inialised. There is no need of sending any data from Robot to Remote Android Controller. The details of implementation of the system has been described in implementation part.

## 5.1 Big Picture

If looked at a large scale, the System can be in four states. These states when looked with their details, can be further divided into some more states. The four states on the bigger scale are described as below:-

### 5.1.1 Idle at Initial Position

The system will be initially in this state only. In this state, the bot will be listening for the Remote Android's broadcasted signal which will set all the parameters like trough no. to go, speed with which the robot should move to the trough etc. Once

initialized the robot will leave this state and will start moving towards the specified trough.

#### 5.1.2 Reaching Trough

After recieving the signal from the Remote zigbee module, the robot will enter into Reachin trough state, In which it will move forward to reach the specified trough. The robot will stay in this stae untill it reaches the specified trough.

#### 5.1.3 Traversing Trough

Once the robot reaches the beginning of the specified trough, It enter into the traversing trough state, and it will start listening to Android phone's command sent to it via bluetooth, which is simultaneously capturing the frames and processing them to generate next command. The system will stay in this state untill it reaches the end of the trough or cuts the detected fruit successfully.

#### 5.1.4 Going back

This state is similar to Reaching trough, in which it has some fixed position where it has to reach, the only difference is, after reaching the destination, It enters into Idle state again, waiting for the next command from the Remote Zigbee module.

### 5.2 Traversing Trough in Detail

The state of traversing the trough begins when the Reaching trough state reaches the trough and ends when either the trough is finished or detected fruit is cut. It can be further divided into three more states. These three states are described as below:-

#### 5.2.1 Searching tomato

This is the initial state. The Android phone will process the Images captured and the robot will keep on moving forward on the trough. If the camera detects a tomato in the frame, The system will leave this state and goto bringing to center state, other it will continue to be in this state untill the end of the trough is reached, then it will skip the bringing to center and cutting the tomato states and will come out of the traversing the trough state directly.

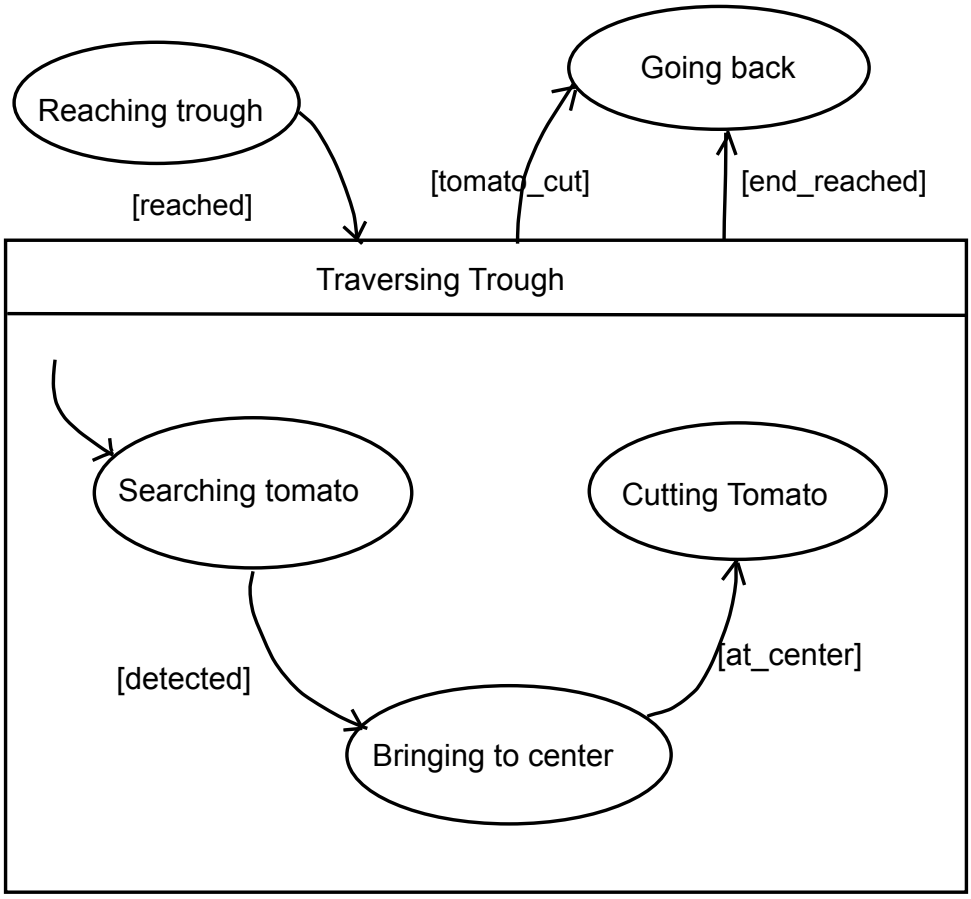


Figure 2: Traversing the trough

#### 5.2.2 Bringing to center

If the tomato is detected on the frame while searching tomato, the system will enter into this state, where it will position the bot and pulley such that the detected tomato comes at the center of the frame.

#### 5.2.3 Cutting Tomato

Once the previous state brought tomato at the center of the camera screen, the bot will enter in this state, in which it will stop moving itself and the pulley, cutter will be turned ON, untill the tomato gets and disappears from the center of the screen. After which the system will leave this state as well as the upper level Traversing the trough state.

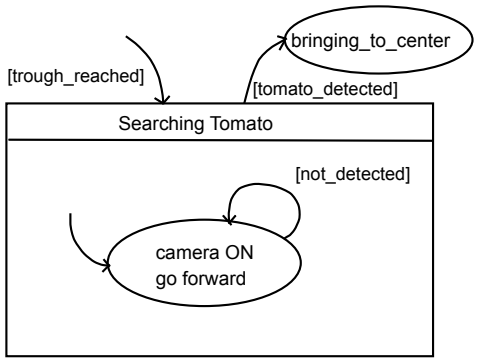


Figure 3: Searching the tomato

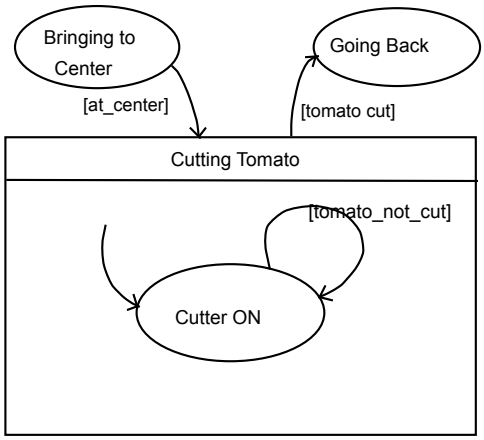


Figure 4: Cutting the tomato

### 5.3 Bringing to center in Detail

Bringing the tomato to the center of the screen involves moving the bot forward and backward, And pulley upward and downward with appropriate amount. Bringing to center is important because the cutter is fixed just upright to the camera and over the pulley and the tomato needs to be infront of the cutter for getting cut. This state comprises of five more states. These three states are described as below:-

#### 5.3.1 Go Forward

This is the initial state. The tomato has just got deteced on the screen but it is still left to the center of x of the screen. Therefore to bring it to the center, the bot needs to move forward. The bot will continue to be in this state untill tomato comes at the center of x or it goes to its right.

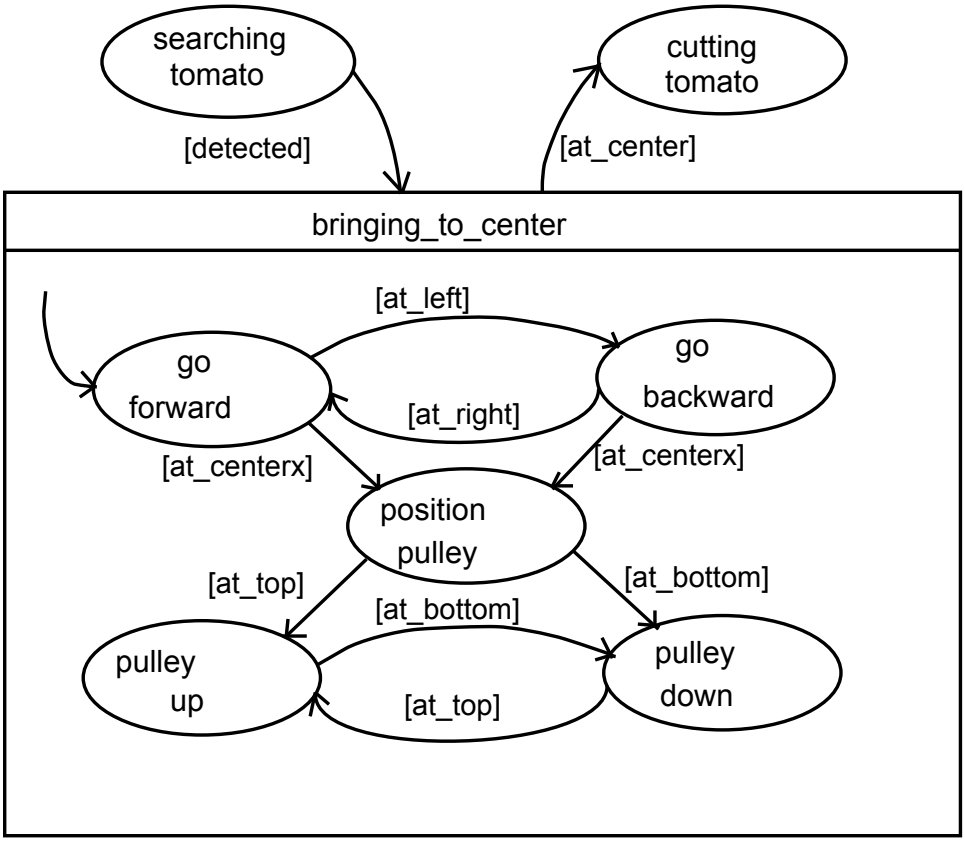


Figure 5: Bringing the tomato to center of the screen

#### 5.3.2 Go Backward

If the tomato has came to right of the center of x of the frame tomato, the system will enter into this state, where to bring it to the center, the bot needs to move backward. The bot will continue to be in this state untill tomato comes at the center of x or it goes to its left again.

#### 5.3.3 Position Pulley

Once the tomato came at the x center of the image, the system will enter into this state, where for the first time it will compare the y location of the detected tomato with the y center of the frame, and accordingly the system will enter either into pulley up or pulley down state.

#### 5.3.4 Pulley Up

If the tomato is above the y center of the frame, the pulley will move up in this state. If it comes at the center the system will enter into the cutting tomato state.

#### 5.3.5 Pulley Down

If the tomato is below the y center of the frame, the pulley will move down in this state. If it comes at the center the system will enter into the cutting tomato state.

## 6. Testing Strategy and Data:

The bot was tested for reaching each trough by giving the instruction through the Zigbee broadcasted packet, and by hanging colored balls in that trough. Energy requirements of the Robot were also tested in the similar situations. The energy consumed by the robot for each test case was recorded. The description of all those tests are are given below:

- **Voltage:** 13.5 V (Average)
- **Current:**
  - When bot is moving : 0.89 A(Approx.)
  - when Cutter or Pulley is ON : 0.80 A (Approx.)
  - When both are ON : 0.90 A (Approx.)
  - Bot moving, cutter and pulley ON : 0.90 A (Approx.)

**Average Current on whole : 0.90 A**

**When speed is 6**

| Trough No. | 1 | 2 | 3 | 4 | 56 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Time (sec) | 42 | 79 | 80 | 80 | 144 | 178 | 212 | 212 |
| Energy (KWH) | 0.1418 | 0.2565 | 0.27 | 0.27 | 0.486 | 0.6008 | 0.7155 | 0.7155 |

**When speed is 7**

| Trough No. | 1 | 2 | 3 | 4 | 56 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Time (sec) | 32 | 56 | 58 | 58 | 102 | 126 | 152 | 152 |
| Energy (KWH) | 0.12 | 0.21 | 0.2175 | 0.2175 | 0.3825 | 0.4725 | 0.57 | 0.57 |

## 7. Resulting Status of Proposed Objectives:-

- Building the mechanical system having a cutter whose position is adjustable on its vertical axis.
  **Status:** Successfully Built.

- Setting up Bluetooth Interface between the Android device and the robot for sending the instructions from Phone to bot.
  **Status:** Successfully Interfaced.

- Processing Captured frames on Android based Phone.
  **Status:** Succesfully Implemented.
  **Problems Faced:** Since opencv is basically implemented in C++ only. We had a hard time to successfully implement its code to Android based Java environment. To implement, possible options were to use, opencv with jni interface on Android or to use javacv for Programming, or to use opencv for Android. There were their own complications in each method, we switched back and forth between these three methods many time untill we figured out how we can do it. In the end we sticked to Opencv for Java and implemented the whole logic of image processing in it.

- The Robot will not cut Partially Ripened Tomatoes.
  **Status:** Unsuccessful. The present System cannot figure out if the tomato is partially ripened or fully ripened.
  **Reasons:** Due to the lack of time, we couldnt implement that since it would have required several trials and errors to detect correct colours correctly.
  **Possible Solution:** To recognize a partially ripened tomatoes, the opencv code should detect the colour of unripened tomato as well and it should check whether the color blobs of the two colours (colour of ripened fruit and unripened fruit) are connected to each other. If yes, then its a partially ripened tomato.

- The bot will cut as many tomatoes as there are on the trough, with nearest tomato first.
  **Status:** Succesfully Implemented. But the bot now considers the biggest tomato first.
  **Reason:** While detecting any colour in a frame, the code will always detect several small false positives due to the bad lighting condition. To ignore those wrongly detected pixels, the code looks for the color blob with largest area

among all the color blobs, since all the false positives are very very small(few pixels). This way, it will always go first to the biggest blob.
**Possible Solution:** What can be done is setting a threshold for the area of the color blobs detected and all the blobs which has their area less than it will be automatically ignored. In the remaining blobs, the one which is at the left most of the frame can then be given priority to implement nearest tomato first.

- The bot will be localized using one camera mounted on the top which will track the position of the bot by recording live frames.
**Status:** Localization is done through programming the bot only. As per the given arena, bot had to move to some fixed troughs, so we hardcoded the distance to travel to reach each trough.

- Proper integration betweeon all the three components of the system, i.e. between the Remote Zigbee Module and the Robot, and between the Robot and the Android Phone.
**Status:** Successfully Done.
**Problems Faced:** It was hard and new for us to integrate codes of Firebird Robot, Android phone, and to integrate opencv code with the Android Phone.

## 8. Conclusion

The implementation of "Tomato Harvester" project has clearly demonstrated us the concepts studied in Embedded Systems Course like time triggered and event triggered activities, deadlines for tasks and how the tasks which appears trivial can give tense moments.

## References

[1] OpenCV 4 Android. http://docs.opencv.org/doc/tutorials/introduction/android-binary-package/.

[2] Android Development. http://developer.android.com/index.html.