CS 684 Project Report

GROUP #3

The Digging Bot

Abhishek Awasthi
133050026

Someshwar Dhayalan
133050078

Samson Swaroop Khess
133050087

Rashmi Rekha Mech
133050089

Contents

1 Introduction

In this document we describe about designing a bot which is a part of a bigger plan to create a fully automated greenhouse for planting vegetables. Our purpose is to create a Bot for digging holes of specific diameter according to the given field plan. The requirement for such a bot is for digging holes to sow seeds or seedlings, at proper distance from each other, and in a proper plan. This document provides an insight into the functionalities expected from the bot, and the possible challenges and risks involved with the design and implementation processes.

1.1 Definitions, Acronyms and Abbreviations

ZigBee: Protocol for Wireless communication PHP: Scripting language for web based applications

2 Problem Statement

To create a Bot for digging holes of specific diameter according to the given field plan.

3 Requirements

3.1 Hardware Requirements

- FireBird V robot
- Linux machine
- Android device with camera and WiFi
- WiFi access point with Internet connectivity
- Robotic arm with a conical head
- Two ZigBee cards

3.2 Functional Requirements

- User will specify the field plan using PHP based interface.
- Field plan consists of location and diameter of the hole.
- The bot will dig holes according to the given field plan.
- The diameter of dug holes will be verified by using image processing.

3.3 Non Functional Requirements

- Intuitive GUI design.
- WiFi Network

3.4 Design Constraints

- With the current digging head, the diameters can be varied from 2cm to 6cm only.
- The bot is be able to dig holes only in soft soil similar to the one present in the greenhouse.
- Users specify only the diameter. The depth is dug in proportion to it.
- Drill used is conical in shape and so are the holes.

4 Implementation

Our problem can be broken down into five sub-problems.

- Determining the location where the hole is to be dug.
- Digging hole and controlling its diameter.
- Capturing the image of hole.
- Doing image processing and determining the diameter.
- Communication between different components

4.1 System Architecture

We have a 3 tier system architecture. The main part of our system runs on a computer system, which is connected to the bot using a Zigbee module and connected to the android camera. This is the core of our system and its purpose is to read the field plan, instructs the bot to reach the position for digging, and also instructs the camera to take picture on which it performs image processing and determine the diameter. The state chard of the system is depicted in Figure 1.

4.2 Firebird Bot system Architecture

We have used the available modules provided to us to simplify our implementation.
There are multiple layers of modules, each supporting well defined primitives. Role of each module is summarized below:

- **Firebird**: This refers to hardware layer of FireBird Bot. Hardware is controlled by changing values in command and status registers.
- **HAL Layer**: Hardware abstraction layer (HAL) module hides hardware registers. It offers typical driver like primitives. init≺Device≻() API initializes the hardware of that device. Table 1 lists all HAL primitives and their use.
- **ROM Filesystem**: This module mimics ROM based file system in limited manner.It redirects standard C file streams stdin and stdout to ZigBee. Standard I/O functions like printf(), putchar(), etc. send data over ZigBee to remote console.This simplifies data transmission over ZigBee as well as helps in debugging Bot code. scanf() and getchar() functions read from ZigBee. This allows to accept arbitrary data from remote machine. Bot can accept arena map files at runtime. AVR C library does not support files directly. We have provided support for compiled-in files using predefined file handles. File handle named MAP FILE can be used to read from compiled-in read-only map file using standard buffered I/O functions like fscanf(MAP FILE, ...).
- **White Line Follower**: This module supports white line related operations.

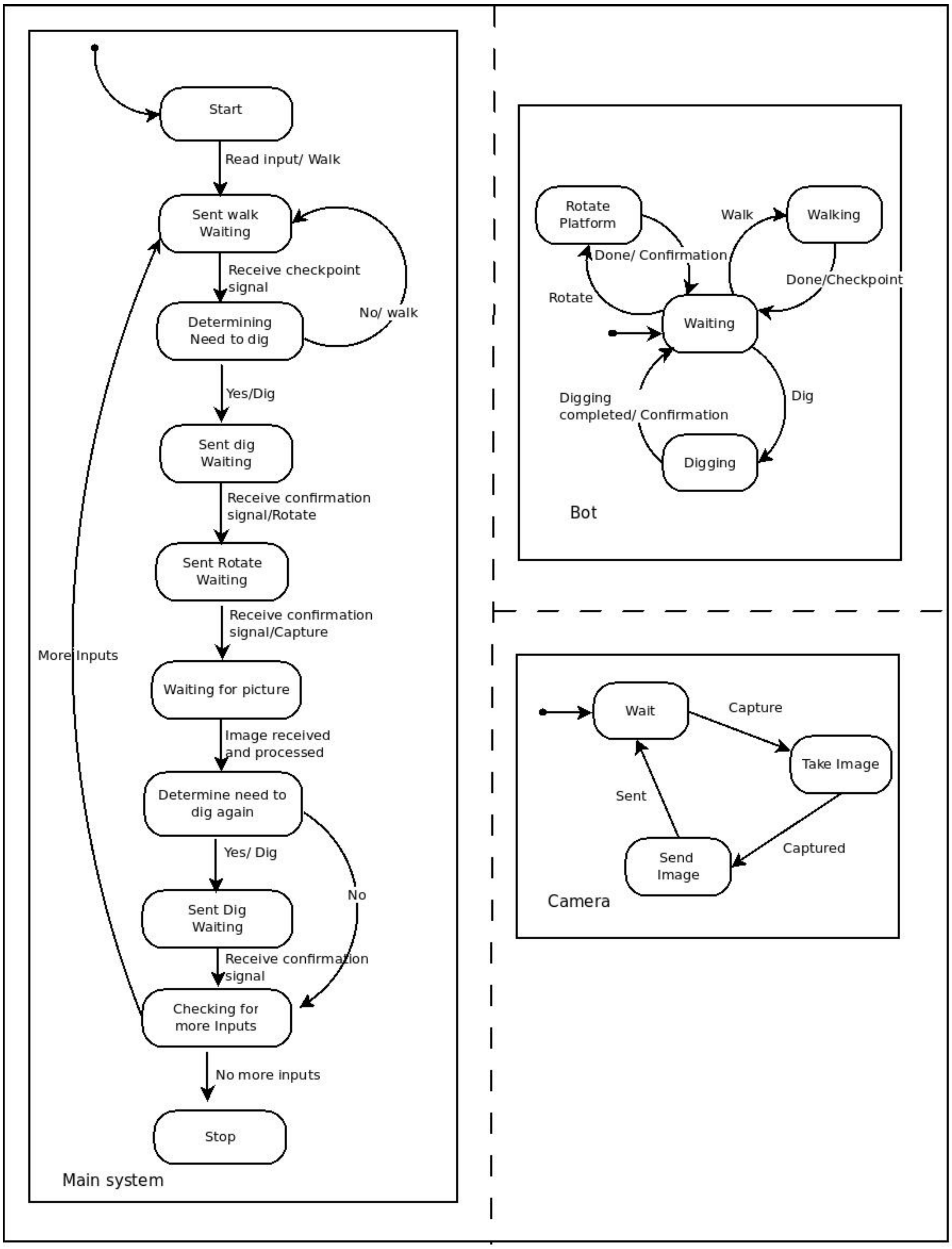

Figure 1: State-chart diagram of the system

4.3 Interface

We have a PHP based user interface which is a graphical representation of greenhouse arena. At each checkpoint (which are 20cm apart), we have a dropdown menu which the user can set to a diameter if he wants to dig a hole there with diameter. For skipping the checkpoint, the user has to simply keep the dropdown blank.

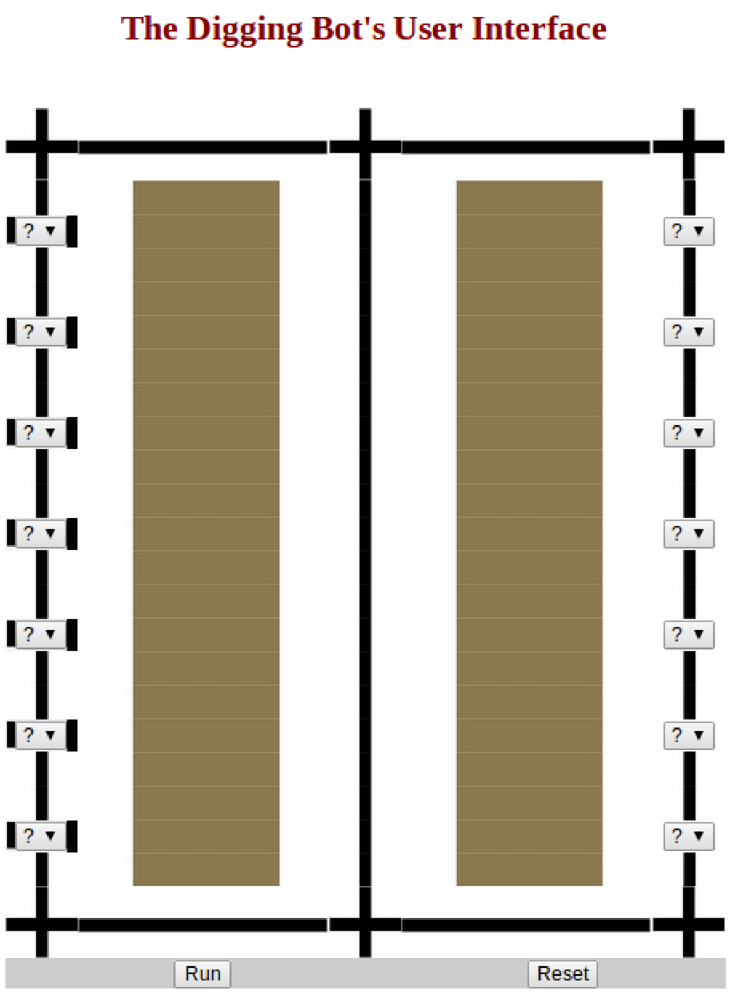

Figure 2: PHP based user interface

4.4 Navigation

The arena we are using is a model of greenhouse, with troughs surrounded by path. We have drawn white-line around the path for the bot to move, following it. Then we have drawn checkpoints at 20cm distance from each other. The bot can detect these checkpoints and the system can determine the exact location of the bot. Moreover, the checkpoints are used as possible positions where digging has to be done.

Whenever the bot encounters a checkpoint, it sends a signal to the central processing system. The system then decides whether hole has to be dug at that location or not. If yes, the system sends a 'dig' command to the bot, and if no, then it sends 'walk' command.

4.5 Digging and controlling Diameter

Our bot has a digging arm which is controlled by two servo motors. There is a conical digging head on the end of the arm. Controlling the downward movement of the digging arm, we determine how deep the head moves, and as its diameter is proportional to depth, we can control the diameter. The two servo motors are synchronized with each other in such a way that the head always hits the soil perpendicularly.

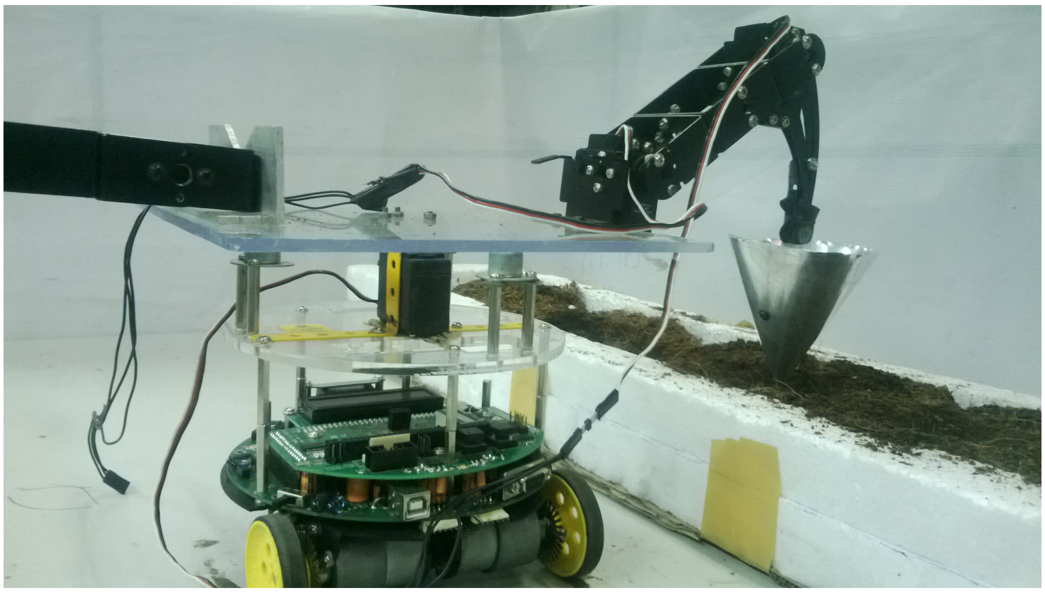

Figure 3: Final System (HardWare)

4.6 Capturing the image

We are using an android device with camera and wifi to capture images. The camera is mounted on the bot and after digging is complete, the bot platform can be turned by 180 degrees using a third servo motor, such that camera is directly above the hole. The android device has an IP webcam installed which can be remotely controlled from a remote site.

4.7 Image processing and determining diameter

We are using contour detection based algorithm to detect the holes and bounding the hole in a circular box. The best possible circular box is used to determine the the diameter of the hole.

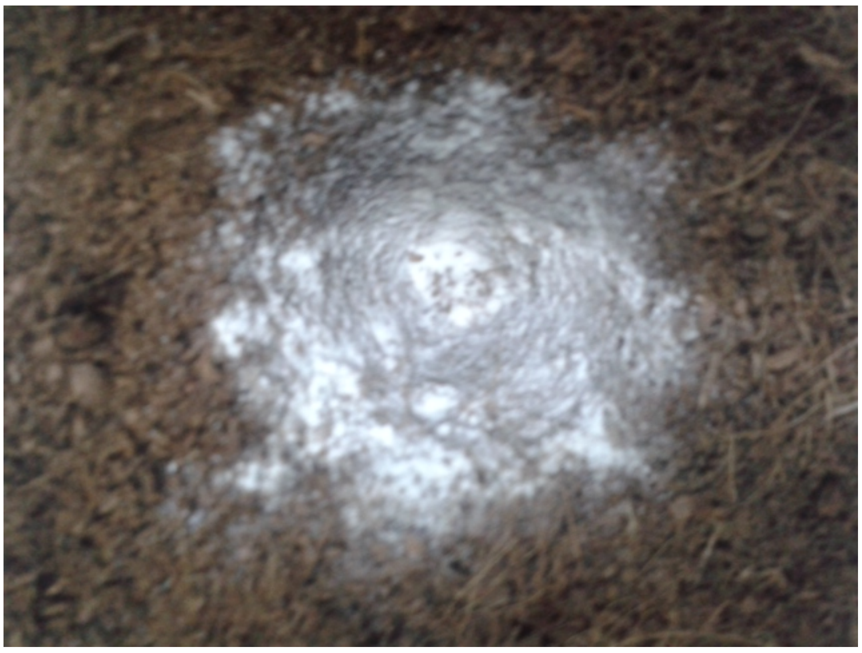

Figure 4: Dug hole

4.8 Communication between components

The control program, which is running on a computer, and the bot, communicate using Zigbee wireless module. All the communication is done one byte at a time, and we send and receive commands and acknowledgements using this.
The communication between the control program and camera happens through

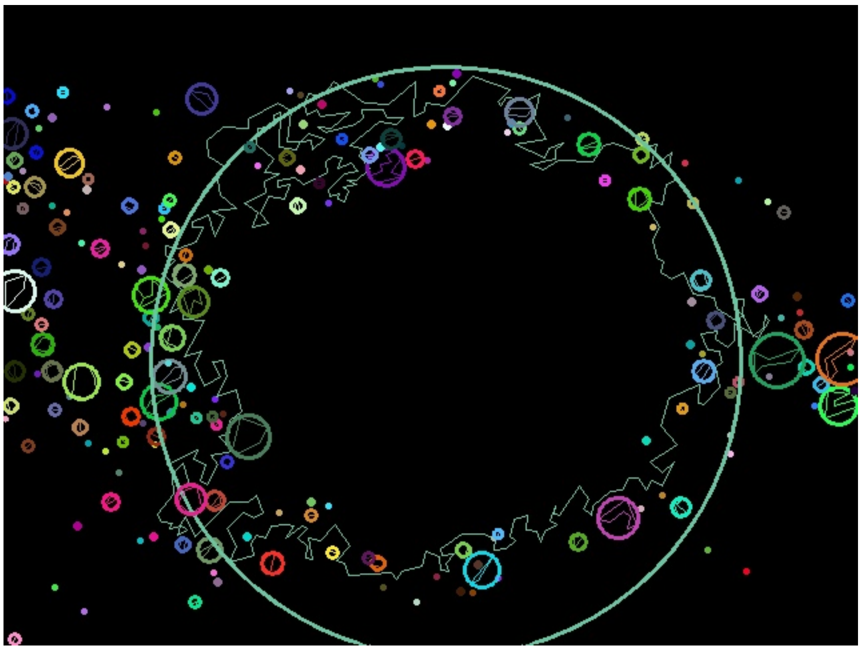
Figure 5: Detected hole.

Wifi network. Both the system and camera are connected to the same network, and the server runs on the android device, which is accessed by our computer and image is received.

5 Testing

For testing, we were given a flex with the ground plan of the arena printed on it. We made troughs out of thermocol and filled it with soil. We tested our bot on different types of soil. When the soil was hard, we found it difficult to dig. Also when the soil was not moist, the hole was collapsing.
The type of soil that is used in greenhouse, coconut coir, was appropriate as it was soft and non-sticky.

6 Design Challenges and Open Issues

6.1 Hole detecting using image processing

As the soil we are using is very dark in color, so distinguishing the hole from such background is a very challenging job. First we tried hole detection using openCV based circle finding algorithm. But it requires that the dug hole must be almost perfect circle, which is rarely possible in real time execution. Then we implemented contour finding algorithm. It searches contours, so also able to detect holes which are not perfect circles.

6.2 Problems with Interrupt handling

We used Zigbee interface to send commands to the bot, but when we called any function to execute the command, till that function was being executed, all the interrupts were getting blocked. It was causing problems when we were trying to move the bot for a particular distance using position encoders. After detecting this bug in our logic, we moved the function call outside the ISR, and set a variable which was used to call the function after ISR finished executing.

6.3 Image acquisition using wireless camera

Handling Wireless camera was difficult as it needed to be tuned whenever the bot was moving. A better and efficient option was to use an Android based device for capturing images as several Android apps are readily available.

6.4 Problems in synchronizing DC motors

Given the same power as input, the two DC motors were rotating in different speeds. This may be because of Electrical or Mechanical imperfections. So we had to calibrate the power for DC motors so that their velocities can be matched.

7 Future Work

7.1 Implementing rotating drill

Rotating drill can be used so that the bot can dig holes on even harder surfaces.

7.2 Adding the seeding feature

Digging with one bot and seeding with another is not very cost effective. So we can also add seeding feature. Different size of seeds can be sown in the hole of appropriate size.

8 Conclusion

References

[1] E-yantra website `http://www.e-yantra.org`

[2] Firebird v atmega2560 robotic research platform hardware manual. IIT Bombay & NEX Robotics Pvt. Ltd.