Dancing E-Bugs

Video Based Multi-Vehicle Systems
Control over Wireless Networks

David McKechnie
Supervisor: Ahmet Sekercioglu
Semester 2, 2008
This project considers a concept for an open-day presentation in which a number of wheeled robots (the E-Bugs) are controlled in an interactive manner: they ‘dance’. The aim of is to create a framework, including hardware and software, for this system.

The E-Bugs are CD sized, battery operated with an Atmel ATmega128 microprocessor running Micrium μC/OS II. Control and User Interface are via a desktop or laptop computer running software authored in Visual C# 2008. The Matlab Image Acquisition Toolbox is used to capture images from a standard webcam, and image processing also occurs in the Matlab environment. The E-Bug recognition algorithm uses a combination of level thresholding and knowledge of the expected shape, and can position shapes at up to 25 fps.

Wireless communication utilizes the Monash Serial Packet Radio board at 433 MHz; with a simple ALOHA style multiple access algorithm. Communication is achieved at 19200 Baud.
6.2 Software ......................................................................................................................... 21
   6.2.1 Communications handling ....................................................................................... 21
   6.2.2 Motor Control .......................................................................................................... 22
6.3 Performance .................................................................................................................. 22
7 Conclusion ....................................................................................................................... 24
   7.1 Interpretation of Results .............................................................................................. 24
   7.2 Recommendations for further work ........................................................................... 24
8 References ....................................................................................................................... 25
Appendices ......................................................................................................................... 26
   A. Visual C# Source Code ................................................................................................. 26
      i. Form1.cs .................................................................................................................. 26
      ii. Form1.Designer.cs ............................................................................................... 35
      iii. Program.cs ............................................................................................................ 47
   B. Matlab Source Code ..................................................................................................... 47
      i. bugfinder3.m ........................................................................................................... 47
   C. C Source Code (E-Bug) ............................................................................................... 50
      i. uCTest.c .................................................................................................................. 50
   D. Monash Packet Radio Schematic ................................................................................. 56
   E. Sens-r main board schematic .................................................................................... 57
   F. Boost Power Supply Schematic .................................................................................. 58

List of Figures

Figure 1 - The E-Bug System ............................................................................................... 4
Figure 2 - The original E-Bug pattern ................................................................................. 5
Figure 3 - Sample Thresholded Images .............................................................................. 6
Figure 4 - Revised E-Bug pattern ....................................................................................... 7
Figure 5 - Final E-Bug Pattern ........................................................................................... 7
Figure 6 - Searching for an E-Bug Dimensions ................................................................. 8
Figure 7 - Midpoint Circle Algorithm ............................................................................... 9
Figure 8 - Bugfinder3 Performance .................................................................................. 10
Figure 9 - Visual C# Software structure .......................................................................... 11
Figure 10 - A Typical configuration screen ..................................................................... 13
Figure 11 - Normal Running conditions ......................................................................... 14
Figure 12 - Monash Serial Packet Radio ......................................................................... 15
Figure 13 - Packet Structure ............................................................................................ 15
Figure 14 - Communications Sequence ........................................................................... 17
Figure 15 - Communication Retries .................................................................................. 18
Figure 16 - An E-Bug

Figure 17 - Tamiya Wheels and Gearbox (left and centre) and Pololu ball caster contact point (right)

Figure 18 - E-Bug software overview

Figure 19 - Travel Distances

Figure 20 - Turn Distances
1. Introduction

1.1 Aim

The aim of this project is to create a basic framework for the ‘Dancing E-Bugs’ open day concept. A number of wheeled robots (hereafter referred to as E-Bugs) are to be controlled wirelessly from a central location, like a laptop or desktop computer. A webcam is to be mounted on the ceiling, allowing this computer to position the E-Bugs in real-time. The E-Bugs are to use a Real Time Operating System (RTOS) in a cheap microprocessor, be battery operated with a regulated power supply, and utilize simple serial communications with a shared medium protocol. The PC software is to be implemented using a combination of standard Windows Forms GUI (Graphical User Interface) front-end, and use Mathworks Matlab for image processing in the back-end.

This project is not specifically concerned with optimizing the performance of the system. Rather, it is an effort to create the basic building blocks which can be further optimized in future undergraduate projects.

1.2 Basis for investigating this topic

The system is designed to incorporate many areas of study encountered in an undergraduate E&CSE course at Monash University. These include, but are not limited to:

- Desktop software design (Visual C#)
- Engineering software design (Matlab / Image acquisition and processing)
- Switching electronics
- Wireless communications
- Embedded software design, including Real Time Operating Systems concepts
- Control algorithms and theory

As such, it is an excellent tool as an Open Day presentation, and highlights many of the concepts learnt in the course in an engaging and interactive manner. The project also serves as a starting point for many more undergraduate thesis projects, which could concentrate on a specific aspect of the setup in more detail, rather than considering the system as a whole.

1.3 Report Structure

This report begins with a summary of the system, with an overview describing what has been achieved, and how each part fits into the whole. This is followed by an in depth look at four subsystems: the image acquisition system, the user interface, the wireless communication system and the E-Bugs themselves. Each area will be discussed both in terms of software, and hardware when appropriate. These sections will include both descriptions of the system in its final state, as well as a discussion of the development and testing of each section. Finally a conclusion will be presented summarizing key findings and recommendations for future work.
1.4 Background information

This thesis is largely inventive and open ended, and as such, no body of knowledge exists to reference as exact material related to this subject. However, similar projects have been undertaken before, notably the HoTDEC project at University of Illinois at Urbana-Champagne [1].

That project differed from this one in the use of hover cars, rather than the wheeled robots envisaged for this project. It was also completed on a much larger budget than dedicated to this one, and employs vastly superior processing power on each of the nodes. There are, however, many aspects that can be used as starting points, notably in the design of a symbol used for recognition, and in the video detection algorithm.
2 System Summary

2.1 Overview

This project is comprised of three distinct components, most easily separated by programming language. They are

i. An image processing system, written in Matlab. This part of the system takes in raw data from a USB webcam, processes it, and provides an array output containing the pixel co-ordinates of any bug shapes detected, the radius of each shape, and the rotation.

ii. PC communications, control and user interface software, written in Visual C# 2008. Communication from this part of the software to Matlab is via the Component Object Model interface available for Matlab. Both software items are run concurrently, with the Matlab server slaved to the C# software, which executes commands remotely. This software makes use of extensive use of Windows .Net runtime libraries.

iii. The E-Bugs. This includes software written for Micrium μC/OS II (C / RTOS) to handle communications and control the motor drive, as well as the hardware design. The E-Bugs are created from a stack of CDs separated by metal stand-offs, with a Tamiya gearbox and motor set. The design also includes a 12V Boost type power supply, in order to maintain near constant speed and turning as the battery voltage drops with use.

An overview of the system is presented in Figure 1.

2.2 Design Assumptions

- The video feed will be relatively clear of objects other than the E-Bugs. Although some ability to discard invalid shapes is included, the video acquisition software should usually be provided with clean data.
- Lighting conditions will be consistent, both throughout the field of view, and over time. Recalibration will be necessary if lighting conditions change dramatically.
- Precision control of E-Bugs is less important than interactivity and speed.
- The camera will be mounted to the ceiling at a fixed height of around 5 – 10 metres.
- Video processing of 25 frames per second will be sufficient for accurate positioning.
Figure 1 - The E-Bug System

a) Pictorial Overview
b) Block Diagram
3 Image Processing System

A crucial component is the image processing routine, and the final source code is provided in Appendix B: Matlab Source Code (page 47). The software is run in two parts. A main Matlab function, `bugfinder3.m`, is used to process images and provide output. Additional to this main loop, there are several one-off commands used to set up the environment, issued directly from the C# environment (see section 4: Graphical User Interface and Control Software, page 11).

Development occurred in stages. Initially, investigation was made into possible alternatives for acquiring video. Methods considered included:

- Matlab Image Acquisition Toolbox
- National Instruments Labview
- Windows DirectX
- Hardware based image acquisition (FPGA based)

Of these, the Matlab Image Acquisition Toolbox provided the fastest development time, and as Monash already has Matlab completely licensed, at very low cost. Labview was strongly considered for some time as the author has experience with that system, but the NI Vision component requires additional licensing which the university does not have. DirectX based capturing appears a reasonable choice, but requires more knowledge of the underlying hardware / driver layer in order to display live video previews etc., whereas this functionality is already available in Matlab with no custom code required. Finally, hardware based processing was considered for use in the circumstance where a PC was unable to process images at a suitable frame rate. This ultimately proved unnecessary.

3.1 Bugfinder 1.1

The initial version of the software was capable of detecting shapes under highly controlled circumstances. It operated in the following way:

i. The 320x240 image is intensity thresholded to provide a Boolean array showing only the white border around each E-Bug pattern.

ii. The Boolean array is searched in every pixel from left to right at vertical intervals of 20 pixels (it is assumed the radius of each E-Bug is at least 20 pixels).

iii. A pair of white borders ‘close’ to each other (within 30 pixels) is taken to be the entry and exit of an E-Bug symbol.
iv. The mean of these two borders is the x position of the bug. From the midpoint, a search is conducted up and down to find the top and bottom borders. The mean of those points is the y position of the bug, while the distance between the two y borders is the diameter.

v. Comparisons are made between the midpoint of this bug, its diameter, and the midpoint of all other already known bugs to prevent double-detection.

vi. A similar search is undertaken over the interior of the bug to find the black dot, using a different intensity threshold, highlighting the black dot against the solid coloured background. Once found, the angle of rotation can be calculated.

This system had several drawbacks. Firstly, it was highly susceptible to false detecting foreign objects in the field of view. Worse, it had a tendency to crash when detecting false positives, as a search would be conducted to find the top and bottom borders that could accidentally search out of the bounds of the image.

Provided a clean image was input, it was also slow for at least two reasons. The main search grid, although on a coarse vertical grid, was through every pixel horizontally. Additionally, searching the interior of the shape for the black dot required many square root calculations (to find the expected extent of the circle), as did comparing the midpoint to prevent double detection. The software was not particularly scalable, and slowed down dramatically as the number of bugs increased.

Finally, although more a problem to do with the pattern (Figure 2) than the software, angle detection was very coarse. Rotation was correct ±20° or more, which was unacceptable.

3.2 Bugfinder 1.2

The revised software had many advantages, although still lacked in performance. A change was made to the bug pattern (Figure 4) after observing that the previous version required very careful fine tuning of the image light thresholds, and a more robust solution was desirable. As such, the pattern was inverted, though this made avoiding the dot when attempting to detect outside edges more difficult. To overcome this, the vertical search grid was reduced to 10 pixels, in order to get a minimum of two search lines across each shape, of which at least one would not include the dot. Figure 3 shows sample thresholds used in Bugfinder 1.2. The left pattern is used first to find probable bug locations and later, to find the angle of each bug. The right pattern is used to find the pixel coordinates and diameter of each bug.
The basic loop structure for the revised software was the same, with extra error handling and checking of search conditions to prevent overflows and searches out-of-bounds. This software performed much more reliably than the original version, but with the same concerns about performance. As expected, angle detection was improved remarkably, to ±10°.

3.3 Bugfinder 3.0

The final bug pattern is shown above. The outside diameter is 12 cm (CD sized), the inside is 9 cm. The radial stripe is at 40% opacity and is 1.5 cm wide. The software is explained in detail below, and differs substantially from previous versions. With the experience gained writing the first and revised versions, an attempt was made to create much faster software from scratch, avoiding trigonometric functions and square root functions where possible. Additionally, the pattern was modified to allow the ‘dot’ (now a line) to be thresholded out when searching for the border, and then a different threshold used to find the stripe and hence the angle.

3.3.1 Finding the inside of the circle
A ‘black image’ is created via intensity threshold, highlighting only the interior of the pattern as a white circle. This is then re-sampled, taking every 20th pixel in x and y directions to create a smaller image for searching. Ideally, each E-Bug will only occupy one pixel in this new image.
A search is run to find white pixels, indicating a bug. After finding a white pixel, a search is performed on the full resolution, thresholded image, to find the left and right borders, and the lower border. This is shown graphically in Figure 6. These three points can be used in the following determinant equation to determine the radius ($r$) and centrepoint ($x$, $y$).

$$\begin{vmatrix}
  x^2 + y^2 & x & y & 1 \\
x_1^2 + y_1^2 & x_1 & y_1 & 1 \\
x_2^2 + y_2^2 & x_2 & y_2 & 1 \\
x_3^2 + y_3^2 & x_3 & y_3 & 1
\end{vmatrix} = 0$$

Equation 1

where $x_n$, $y_n$ are the $x$ and $y$ coordinates of each point found in the search. From [2], Equation 1 can be rearranged in conjunction with the standard equation for a circle (Equation 3) to yield:

$$x = \frac{M_{12}}{2 \times M_{11}}, \quad y = \frac{-M_{13}}{2 \times M_{11}}, \quad r = \sqrt{x^2 + y^2 + \frac{M_{14}}{M_{11}}}$$

Equation 2

Where $M_{ij}$ in Equation 2 is the cofactor of element $(i,j)$. For instance, $M_{13}$ allows for a quick calculation to find the centre and radius of the bug. The radius result is then checked to confirm it is in the allowable range before continuing to find the angle.

3.3.2 Midpoint Circle Algorithm

A raster graphics technique called the Midpoint Circle Algorithm from [3] is utilized to search in a circle around the midpoint at 60% of the radial size, to find the stripe and hence, the angle. This algorithm avoids the use of any trigonometric or square root calculations when searching in a circle, greatly increasing the speed of the system.
The algorithm calculates a 1/8 segment of the circle’s pixel coordinates, which can then be mapped through mirroring to give the coordinates of every pixel that lies on the circle. The algorithm functions by assuming that every step requires one movement in the x direction, and may or may not require a downward movement in the y direction. We start with the function for a circle:

\[ F(x, y) = x^2 + y^2 - r^2 \]

Equation 3

\( F \) in Equation 3 is

- Zero on the circle
- Positive outside the circle
- Negative inside the circle

An iteration of the algorithm has two possible outcomes for \( y \): either it remains the same, or it moves down one. Figure 7 shows this graphically. Starting in the top left, a midpoint \( M_1 \) is chosen to lie between the next two possible points. Evaluating \( F(M_1) \) yields a negative number, which means that \( M_1 \) is inside the circle and no change in \( y \) is necessary. Repeating this procedure at \( M_2 \) yields a positive response, which means \( M_2 \) is outside the circle and a change in \( y \) is necessary. Thus a full circle can be mapped without using trigonometric nor square root functions.

Each point on the circle is then checked to see if it is black or white, with an intensity threshold set to reveal the difference between the stripe and the white background. The black pixels are collected and the mean value used to calculate the angle of the bug using simple trigonometry.

### 3.3.3 Preventing double detection

Rather than the cumbersome method of measuring distances between previously detected bugs, BugFinder3 blanks out a 3x3 grid in the low resolution search grid image, centered on the location of a successfully characterized bug. This will prevent double detection by eliminating the bug from the search pattern, although there is a possibility that if two bugs are positioned too close to one other, only one will be detected. Testing revealed this to be a relatively rare occurrence.
3.3.4 Performance

Testing revealed an ability to detect 5 objects correctly at 24 frames per second, as shown in Figure 8, on a 2 GHz Intel Core 2 Duo laptop. Changing the number of objects impacts the speed with which recognition takes place, though with 9 objects in the field the speed of detection is still satisfactory. It is expected that the system will ultimately use 8 or more E-Bugs, and the image recognition system should be able to cope with this target. Angle detection remains accurate to ±10°.

Integrating the Matlab script with the C# program decreases the frame rate due to the overhead communicating between the two environments; however testing found the rate to be acceptable, and seemingly ‘real-time’.

3.3.5 Considerations when capturing repeated images in Matlab

The Matlab command

getframe(vid)

is used to take a static frame from a video object that has previously been opened, using

vid = videoinput(…)

with appropriate arguments. Normally, this process can take several seconds, which is not acceptable given the timing constraints. To avoid this delay, the video must be started and configured for manual triggering. The command

triggerconfig(vid, ‘manual’)

will open the video resource vid and prepare it for capture, but will not actually record any images. Additionally, the camera driver will tend to autofocus regularly, which upsets the detection process. Forcing the camera to manual focusing will avoid this, using

src_vid = getselectedsource(vid) and set(src_vid, ‘Focusmode’, ‘manual’).

This process must occur before the main loop is run to capture and process images.
4 Graphical User Interface and Control Software

The main program, with which the user interacts, is written in Visual C# 2008 Express Edition, and full source code is presented in Appendix Visual C# Source Code (page 26). Although the image capture and processing system was written within Matlab, it is desirable to have other elements written in an environment that more easily allows for object oriented programming (Matlab has object oriented coding support, but it is rudimentary) and that also allowed for simple multi-threading: Visual C# fulfills these requirements. It also has very tight integration with the Microsoft .Net framework, which exposes a massive library of resources allowing rapid development of Windows forms applications, as well as communication to other software (Matlab) and access to serial ports etc.

Additionally, Visual C# allows extensive use of custom events and delegates, which simplify control of a Windows form from multiple threads. Windows forms displays are not inherently thread safe, so care has to be taken when information is being updated from multiple sources. Writing a safe, multithreaded Windows forms application in C++ application, for instance, is more difficult.

Communication between C# and Matlab is achieved via the Component Object Model, which exposes some methods allowing remote execution of Matlab code from within another programming environment [4].

4.1 Program Structure

Figure 9 - Visual C# Software structure
The structure of the eBug Control software is shown in Figure 9. This diagram does not include all of the methods and objects in the software, but a subset of important items.

Initially, only Form1 is loaded, and it loads the _MLAppClass object (causing an instance of the Matlab Server to be opened in the background), as well as creating the _serialPort object. After the user has configured the serial interface and camera using the GUI, a RobotBug bugs[] array is created, and the bgWrkrGetCoords object begins running.

RobotBugs is a custom class which includes as members all the appropriate state information about each E-Bug (x position, y position, rotation, ready state etc) as well as a queue of commands to be sent to the E-Bug as it becomes ready. The methods of a RobotBug object are used to communicate: for instance, after assigning bug[o].destination, bug[o].length_index and bug[o].direction, the method bug[o].sendCommand() can be called. This method will compile the bug’s members into a well formed packet, calculate a checksum, and interact with the _serialPort object to send the data.

bgWrkrGetCoords is an asynchronous thread which instructs the Matlab server to perform various duties. For instance,

```matlab
_MLAppClass.Execute('bugs = bugfinder3(getshapshot(vid),100,100,20);');
```

instructs the Matlab server to run the command to take a new image, process it and store the results in a Matlab variable bugs. Later,

```matlab
_MLAppClass.GetFullMatrix('temp', "base", ref XReal, ref XImag);
```

returns the variable ‘temp’, from Matlab workspace ‘base’ and puts the results in local variables XReal and XImag. This thread runs continuously, and updates the bugs array with new information as it becomes available.

There are several aspects of the program not shown in Figure 9, omitted for clarity. The _serialPort object is guarded by a semaphore, which prevents multiple bug objects from attempting to send data at the same time. An acknowledge semaphore also exists, used to signal to a waiting bug that data was successfully sent to its destination. Missing also are the procedures to update the GUI, which must occur every time the bgWrkGetCoords object retrieves new data. This occurs by means of a delegate, where the background worker attempts to update the form (which is unsafe), and is intercepted. The update is then delegated to the form thread itself, which will run on the next context switch. The form is notified only that an update is required: it then retrieves all the values directly from the robotBug array.

Some user interface controls interface with the _MLAppClass object: controls to pan or zoom the camera, for instance, send ‘one off’ commands to make appropriate adjustments. Additionally, several commands are sent prior to opening the camera for capture, as mentioned in section 3.3.5.

### 4.2 User Interface

The User Interface comprises of three main sections:

- The configuration tab, where setup occurs before normal operation can begin.
- Various Matlab output windows (live video preview, image calibration display panes).
» The Run tab, where interactive control of the E-Bugs takes place (normal operation).

4.2.1 The Configuration Screen

Figure 10 shows the configuration panel, after a typical setup and calibration has been performed. Clicking the Start Video Preview button will cause the Video Preview pane in the top right corner to appear, streaming live video from the camera. This can be adjusted using the Pan, Tilt and Zoom commands to achieve the desired frame.

Calibration levels must be set to for the image processing software to use. The BlackCal image should be entirely black, save the outline of the E-Bug. The stripe showing the angle should also be clearly visible. This image is searched to find candidates for recognition, and later used to determine the angle. The WhiteCal image can have extraneous noise in regions outside the circular pattern, but the circle must be clearly defined, and the angle stripe should not be visible. This image is used to find the edges of the pattern, and hence determine its position and size.

4.2.2 The Run Screen

The controls on the Run screen only become available once a COM port has been successfully opened, and a camera preview successfully started. There are two ways to control an E-Bug:

» Manual Control. This allows ‘remote control car’ style driving of the E-Bug, with turns specified in degrees and distances specified in cm.

» Automatic Control. This allows a destination to be set using the green crosshairs on the output plot, and the robot will perform a turn and a straight line maneuver to relocate itself automatically to the new position.
Figure 11 shows the controls of a typical Run screen. Note that the calibration panes have been closed: re-calibration is currently not available once processing has begun. This means the program must be restarted if dramatic changes in lighting occur while it is being used.

4.3 Stability and performance

4.3.1 Stability

The software is stable under most circumstances. The only consistent way to cause a fault is to close the Matlab server window before closing the eBug control panel, which will cause a crash when the _MLAppClass object is next called. It is possible to run the Matlab server in ‘silent’ mode, to avoid this problem, but for testing and development it is convenient to have it visible.

The software works under constant lighting conditions and successfully detects and characterizes the bug patterns as desired, at an acceptable frame rate. ‘Open loop’ manual control of the E-Bug is reasonably reliable, and around 80% of commands are responded to. Automatic control of the E-Bug is very intermittent. The fault centers on the ability to receive the ‘Ready’ status of an E-Bug, which is transmitted when it reaches its destination. This transmission is rarely received correctly when running in full automatic mode, and so the system will stall, waiting for this response.

The ‘Square’ button in the interface queues a series of commands that should cause the E-Bug to drive in a square pattern. Although the queuing concept has been demonstrated successfully, actually driving in a square, as desired, has never been observed due to the aforementioned stalling problem.
5 Wireless Communications

Central to the wireless communication of the system is the Monash 433 MHz Serial Packet Radio, shown in Figure 12. The computer side communications use an unmodified version of this circuit, running at 19200 baud. On the E-Bug side, the MAX232 RS-232 driver has been removed and the TTL level signals wired directly to USART1 on the ATmega128. The boards have been tested with a working range of 10 metres, though this performance is not consistent.

Late in testing it was discovered that the quality of transmission is massively dependent on the board location, and this presents a problem when the boards are used on a wheeled vehicle. Several attempts were made to overcome the instability in reception, but with little success. The problem manifests as a signal that is transmitted and received, but malformed in some way. For instance, a typical corruption is to turn the hex sequence 0xAA into 0xEB or similar. When the corruption occurs, it can be sustained for a lengthy period (10s of seconds), and usually requires a complete stall and reset of communications before it is resolved. Changing the multi-path interference by waving an object around the antennae can help.

Attempts to correct these faults by lowering the transmission speeds were unsuccessful, as the USART on the ATMEGA device is unable to synchronize to the start bits of any transmission below 4800 baud. After extensive testing, 19200 baud was found to provide the most reliable link. The reasons for increased reliability at this speed have not been investigated.

5.1 Packet Structure

The packet structure is shown in Figure 13. The packet length is fixed, regardless of payload, to enable a simpler receive and decode algorithm. The fields are:
» Preamble Byte. The preamble byte is used to mark the start of the transmission. It was found in practice that retransmitting this several times before transmitting the packet increased reliability, and helped avoid the problem outlined above.

» Destination Address. The address of the E-Bug. Although the E-Bug is hard coded with an address in the current version of the software, there is space on the circuit board to accommodate a set of switches, which would be used to set the address in future.

» Source Address. The controlling computer uses a source address of 0xFF.

» Direction. The direction field can have one of 5 different values:
  - FORWARD: oxA. Indicates that the E-Bug should drive forward.
  - PIVOTRIGHT: ox6. Indicates that the E-Bug should turn right.
  - PIVOTLEFT: ox9. Indicates that the E-Bug should turn left.
  - ACK: ox10. Used to acknowledge receipt of a message.
  - RDY: ox11. Used to inform the controller that an E-Bug has finished its current maneuver.

» Length Index. Used in conjunction with the Direction field, this is used to either represent the angle of turn required by sending \(\text{angle}/10 - 1\), with \text{angle} in degrees, or the distance to move forward, by sending \(\text{distance}/5 - 1\), with distance in centimeters.

» Sequence Number. This number is incremented by the control computer with each new message. It is used by the E-Bug to filter out repeat transmission, in the case that an ACK message is lost.

» Checksum. Hashing function output used to filter corrupt packets.

The length index and sequence number fields only have meaning when a movement command is being sent from the control computer to an E-Bug. When an ACK or RDY message is sent, the values of these fields are undefined and contain no useful data (though they are still included in the checksum calculation).

5.2 Communications Protocol

The protocol designed is a simple ALOHA [5] style multiple access protocol. The concept is basic:

1. If data is ready, send it.
2. After sending, if an ACK is not received within some predetermined timeframe, wait a random period and send it again.
3. If you receive a packet, ACK it.

This is implemented in both PC and E-Bug by sending data, then pending a semaphore with a pseudo-random timeout. If the semaphore is posted by the receiving thread upon receipt of an ACK, then the message will not be sent again. However, if the semaphore pend operation times out, a collision is assumed to have occurred (either in the sent data, or the ACK) and the message is retransmitted.

The inclusion of a sequence number allows detection of repeat transmissions in the case of a lost ACK, and the system is simplified somewhat by the recognition that there is no communication between E-Bugs: all communication is from E-Bug to control computer and back. As such, the E-Bug need only keep track of the last sequence number, and compare it to that of any new messages. A message will be acted upon if and only if it has a new sequence number attached to it.
Figure 14 illustrates this system, showing a typical exchange of messages when an E-Bug is commanded to drive forward. Two collisions are assumed to occur, and it can be seen that the behavior of the E-Bug is entirely predictable. It does not double process the command, although it is sent a total of three times.

5.3 Considerations when transmitting from the E-Bug

A problem encountered early in the implementation of this system was the occasional reception of self-broadcast messages when transmitting from the E-Bug. Usually these messages are corrupted, but they can interfere enough to cause fault situations, and need to be avoided. An immediate workaround included disabling the USART receive capability when transmitting, and enabling an interrupt on the TX buffer empty signal. When the buffer is empty, reception can be turned back on.

Surprisingly, this problem does not seem apparent when using the board on a PC. It is surmised that the serial driver is able to detect the malformed signal, and discards it.
5.4 Performance

When the protocol is running normally it performs well, achieving transmission to each E-Bug with acceptably low delay. Figure 15 shows the number of retries in a test setup before a message is successfully delivered and acknowledged. 60 attempts were made in total; with one node attempting to communicate and another generate spurious noise at intervals to simulate collisions.

![Figure 15 - Communication Retries](image)

It can be seen in the results that many of the communications were delivered in at worst, 6 retries. Given the very short transmission time of each packet, that is considered an acceptable delay. There are, however two distinct groups of transmissions: those which transmitted reliably, and those that needed some 'help'. The transmissions with 16 or more retries are those which appeared to be failing until an object was waved next to the antennae, which changes the channel characteristics and can produce a successful transmission. These delayed transmissions cause any of the automatic control procedures to lock up and fail, as mentioned in section 4.3.1. The transmissions in the ‘More’ category can be considered failed, because the code will only retry a fixed number of times before deciding the recipient is offline.
6. E-Bug

The E-Bug main board design is taken from [6], with slight modifications. Primarily, the connection previously used to connect to the infra-red communication board was removed and used instead for wireless communications. Modifications were also made in an attempt to dampen electrical noise in the board, by connecting the bulk copper plane to ground.

6.1 Hardware

6.1.1 Hardware description
The E-Bug is constructed out of a stack of blank DVD discs, separated by metal standoffs. Power supply is from a set of 6 x AA batteries sitting above the gearbox, which drive a 5V regulator for the board operation, as well as a 12V boost power supply positioned at the bottom front of the bug. This 12V rail is used to drive the motors.

The Sens-r main board is located on the bottom DVD, and based on an Atmel ATMega128 running Micrium μC/OS II Real Time Operating System. The board also includes an L293 motor controller.
with PWM. A Monash Packet Radio board is located on the middle DVD, with the RS-232 driver IC removed, as previously mentioned. All connections are routed through the centre of the stack of discs, leading to a very neat design. The top layer is used to hold the bug pattern, which is printed onto plain white paper and attached with plastic contact adhesive film for longevity.

The gearbox used is a Tamiya 70168 Double Gearbox Kit at 115:1 gear ratio, with the motors from the kit removed and replaced with Dick Smith Electronics P9000 motors. These motors are more efficient, lower speed and higher torque. A third contact point on the bottom of the robot is provided by a ½ inch metal ball caster, from Pololu.com.

The boost power supply is constructed using a National Semiconductor LM2587-12 TO-220 switching regulator package, and is capable of producing up to 1.5A @ 12V, over an input voltage range of 4-12V. Using a regulated power supply for the motors makes their behavior substantially more predictable, as there is no droop in the voltage as the batteries run flat.

6.1.2 E-Bug Development
The E-Bug has only undergone one major change during its development, which was an aborted attempt to use optical encoders on the gearbox to provide more accurate movements. The original design included encoders, which were successfully used in [6] with very similar hardware. Although the electronics for the encoders were successfully designed, constructed and verified, integration with the ATMega128 proved more troublesome. An interrupt was successfully triggered from rising edges on the encoders, but running the motors and these interrupts at the same time caused the microprocessor to constantly reset.

Extra steps taken to overcome this problem included the addition of clamping diodes and smoothing capacitance to all motor signals. A hex inverting buffer was also introduced in an attempt to limit the noise sensed by the microprocessor, but the results were no better. Eventually, the encoders were abandoned, and motion is instead controlled using internal timers. This is a somewhat less satisfactory result, but proved acceptable for the desired purpose.

Initial testing was conducted using a direct serial connection for debugging purposes, in conjunction with a set of LEDs available on the board. An external RS232 interface board was created to connect to the USART0 pins available on the board.
6.2 Software

The software is written using in AVR Studio 4 using the WinAVR20080610 GNU GCC implementation for compilation, and is based on version 2.60 of μC/OS II, with AVR port support files by Ole Saether/Julis Luukko [7]. The board is programmed using an Atmel AVR ISPMk2, using 10 wire In System Programming (ISP) [8]. Slight modifications to the ISPMk2 were required, as documented at [9].

![Diagram of E-Bug software overview]

Figure 18 shows the basic layout of the E-Bug software. AVRInit() and OSInit() are called before multi tasking begins, and initialize the ATmega hardware (USART Rx/Tx, PWM etc) and the real time operating system, respectively. A single task is created called TaskStart, which initially serves to create two more tasks, MotorTask and MotorTimer, before becoming the communications handling task.

6.2.1 Communications handling

UCOSISR(USART1_RX_vect) is triggered whenever new data arrives on the USART1 interface. This ISR receives the data and adds it to a global message buffer. If it notices the start of a packet, it moves the data to the front of the queue and upon receiving a complete packet, posts the UartRxSem semaphore.

After starting MotorTask and MotorTimer, the TaskStart task waits on the UartRxSem semaphore until the receive ISR posts it, signaling a complete packet received. It then processes the data out of the global receive buffer, checking the validity of the received message. If it receives a valid message, it sets the appropriate motor control signals, and sends an ACK back to the message author, before returning to wait on the UartRxSem semaphore.
6.2.2 Motor Control

Two tasks are used to control the motors. The first, MotorTask, simply checks a global target setting for each wheel PWM and ramps the wheel PWM with a fixed delay between steps to match the target. In this way, the motors are always soft started and stopped, preventing jolting and excessive noise.

The second task, MotorTimer, controls the targets used by MotorTask. When a new packet is received, the TaskStart task takes the Length_Index field and performs a lookup to find an appropriate delay to yield the required angle/distance. It then copies this value to a global variable and posts the WheelTimerSem semaphore. MotorTimer waits on this semaphore, and when triggered, sets the target PWM to 0xFF for the time set by TaskStart. After this, the MotorTimer task sets the target PWM to 0x0, and sends a RDY message to the control computer, before returning to wait on the WheelTimerSem semaphore.

6.3 Performance

The speed of the E-Bug is estimated to be 30cm/s, which is acceptably quick. Turns are complete in well under 1 second. The accuracy of the E-Bug is surprising given the lack of optical encoders, with an ability to repeat turns to within 10 degrees, and drive forward with an error of only several centimeters/m. Normally these sorts of errors would be intolerable, but because of the closed loop camera system, they can be corrected for when issuing the next instruction.

![Figure 19 - Travel Distances](image)

Figure 19 shows the distance covered by the E-Bug in three attempts over a range of 5cm to 1 metre, at 5cm intervals. This data was used to calibrate the E-Bug lookup table in order to reproduce these movements on command. Figure 20 likewise shows the results used to calibrate the lookup table for turning left and right.
Figure 20 - Turn Distances
7 Conclusion

7.1 Interpretation of Results

Two of the major goals of this project, fast image processing and control of a wireless wheeled vehicle, were thoroughly investigated and suitable software developed. Areas of investigation included:

- Programming a multi threaded Windows Form application
- Interfacing with Matlab from external software
- Fast raster search techniques for analyzing images
- Multiple access wireless communications techniques
- Programming for a Real Time Operating System target

In conclusion, this thesis project successfully delivered:

- A prototype E-Bug, including software for communicating and moving
- Image processing software capable of tracking several targets in real time
- A Windows Forms application for control of the system.

Due to time and budget constraints, multiple E-Bugs could not be constructed in order to verify scalability of the system. Additionally, the Monash Serial Packet Radio does not provide a reliable enough channel to successfully control the system in the desired fashion. Given more time and investigation, an alternative wireless system would be implemented, which would allow an experiment to be conducted to verify the complete operation of the system.

7.2 Recommendations for further work

The system as it stands is a framework for much further work. Immediately, investigation into an alternative wireless system will allow ‘closing of the loop’, and demonstration of the entire control system from top to bottom. Moving forward, more complex control algorithms can be developed allowing more than just point-to-point movement. This would necessitate additional feedback on the E-Bug, in the form of optical encoders or any other local position tracking mechanism, as originally planned in this design.
8 References


Appendices

A. Visual C# Source Code

i. Form1.cs

```csharp
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Threading;
using MLApp;
using System.Drawing.Drawing2D;

namespace eBugControl
{
    public partial class Form1 : Form
    {
        delegate void SetTextCallback();
        delegate void SetDebugCallback(string text);

        MLAppClass _MLAppClass = new MLAppClass(); // Matlab Interface

        const byte FORWARD = 0xA;
        const byte PIVOTRIGHT = 0x6;
        const byte PIVOTLEFT = 0x9;
        const byte ACK = 0x10;
        const byte RDY = 0x11;

        public SerialPort _serialPort = new SerialPort();
        public Semaphore ack_semaphore = new Semaphore(0, 1);
        public Semaphore auto_semaphore = new Semaphore(0, 1);

        public robotBug[] bugs; // A home for all the bugs
        public byte[] recvBuffer = new byte[1024];
        public int recvBufferReadIndex = 0;
        public int recvBufferWriteIndex = 0;

        public Semaphore queue_sem = new Semaphore(1, 1);

        public Form1()
        {
            InitializeComponent(); //Set up and position all the buttons and stuff
        }

        // Runs at form startup
        private void Form1_Load(object sender, EventArgs e)
        {
            foreach (string s in SerialPort.GetPortNames())
            {
                listComPorts.Items.Add(s); // Populate the com port list
            }
            listComPorts.SelectedIndex = 0; // Select the first item in the lists
            listBauds.SelectedIndex = 0;
        }

        // When a user clicks the 'Connect' button
        private void buttonComConnect_Click(object sender, EventArgs e)
        {
            if (_serialPort.IsOpen) // Close any previously opened ports
            {
                comPortDebug.Text += "Closing " + _serialPort.PortName + "\n\n";
                _serialPort.Close();
            }
        }
    }
}
```

```csharp
SerialPort _serialPort = new SerialPort();
_serialPort.PortName = listComPorts.Text; // Set the COM port
_serialPort.BaudRate = int.Parse(listBauds.Text); // And the baud rate
_serialPort.Parity = Parity.None;
_serialPort.DataBits = 8;
_serialPort.StopBits = StopBits.One;
_serialPort.Handshake = Handshake.None;
comPortDebug.Text += "Opening " + _serialPort.PortName + " @ " + _serialPort.BaudRate + "bps\n";
try
{
    _serialPort.Open();
}
catch (IOException)
{
    comPortDebug.Text += "ERR: That COM port doesn’t exist!\n\n";
}
catch (InvalidOperationException)
{
    comPortDebug.Text += "ERR: Com Port already open\n\n";
}
catch (UnauthorizedAccessException)
{
    comPortDebug.Text += "ERR: Com Port in use, access denied\n\n";
}
if (_serialPort.IsOpen) // Everything worked!
{
    comPortDebug.Text += "Open!\n\n";
    if (panelCamControls.Enabled) panelRun.Enabled = true; // Enable the Run panel
    else panelRun.Enabled = false;
}
```

Appendix A: Visual C# Source Code
Form1.cs
private void trackBarDistance_Scroll(object sender, EventArgs e) {
    int value = (trackBarDistance.Value / 5) * 5;
    labelDistanceDisp.Text = value.ToString();
}

private void trackBarAngle_Scroll(object sender, EventArgs e) {
    int value = (trackBarAngle.Value / 10) * 10;
    labelAngleDisp.Text = value.ToString();
}

private void buttonSetBugs_Click(object sender, EventArgs e) {
    bugs = new robotBug[(int)numBugs.Value];
    comboBugSelect.Items.Clear();
    for (byte i = 0; i < numBugs.Value; i++) //Create a bunch of bugs with default values
    {
        bugs[i] = new robotBug();
        bugs[i].Form = this;
        bugs[i].ready = true;
        bugs[i].x = 0;
        bugs[i].y = 0;
        bugs[i].radius = 0;
        bugs[i].srcaddress = 0xFF;
        bugs[i].dstaddress = 0x00;
        bugs[i].sequence = 0x00;
        bugs[i].serialPort = _serialPort; //Point them all at the same serial port
        bugs[i].acksem = ack_semaphore;
        bugs[i].auto_stage = 0;
        comboBugSelect.Items.Add("Bug " + bugs[i].dstaddress);
        comboAutoBugSelect.Items.Add("Bug " + bugs[i].dstaddress);
    }
    //Add the data received event handler
    _serialPort.DataReceived += new SerialDataReceivedEventHandler(_serialPort_DataReceived);
    updateBugStatus(); //Display some bug stats
    comboBugSelect.SelectedIndex = 0;
    comboAutoBugSelect.SelectedIndex = 0;
    groupManual.Enabled = true;
    int[] CalValues = new int[2];
    CalValues[0] = trackBarBlackCal.Value;
    bgWrkerGetCoords.RunWorkerAsync(CalValues); //Start running the thread to grab data from Matlab
    bgWrkerAutoControl.RunWorkerAsync(); //Start the queuing thread
}

//Called whenever a bug changes its status
public void updateBugStatus()
private
{
    if (this.textBoxBugStatus.InvokeRequired)  // If it wasn't called by the right thread
    {
       SetTextCallback d = new SetTextCallback(updateBugStatus);
        this.Invoke(d, new object[] { });  // Tell the right thread that there's work to do
    }
    else
    {
        textBoxBugStatus.Clear();
        textBoxBugStatus.Text += "\RDY?\tX\tY\tT\tAngle\r\n";

        foreach (robotBug _robotBug in bugs)
        {
            textBoxBugStatus.Text += "Bug " + _robotBug.dstaddress + "\t"
            if (_robotBug.ready) textBoxBugStatus.Text += "RDY"
            textBoxBugStatus.Text += "\t" + _robotBug.x.ToString("0.0")
                + "\t" + _robotBug.y.ToString("00.0")
                + "\t" + _robotBug.radius.ToString("00")
                + "\t" + _robotBug.angle.ToString("0.00")
            ;
            textBoxBugStatus.Text += "\r\n";  // Invalidate the drawspace, so it gets repainted
            panelBugSpace.Invalidate();
            foreach (robotBug _robotBug in bugs)
            {
                if (_robotBug.ready) textBoxBugStatus.Text += "RDY"
                textBoxBugStatus.Text += "\t" + _robotBug.x.ToString("0.0")
                    + "\t" + _robotBug.y.ToString("00.0")
                    + "\t" + _robotBug.radius.ToString("00")
                    + "\t" + _robotBug.angle.ToString("0.00")
                ;
                textBoxBugStatus.Text += "\r\n";  // Invalidate the drawspace, so it gets repainted
                panelBugSpace.Invalidate();
            }
        }
    }
}

public void writeRunDebug(string text)
{
    if (this.textRunDebug.InvokeRequired)
    {
        SetDebugCallback d = new SetDebugCallback(writeRunDebug);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        textRunDebug.Text += text;
    }
}

// Manual control buttons
private void buttonForward_Click(object sender, EventArgs e)
{
    int bug_index = int.Parse(comboBugSelect.Text.Substring(4, 1));
    bugs[bug_index].direction = FORWARD;
    bugs[bug_index].length_index = (byte)(byte.Parse(labelDistanceDisp.Text) / 5 - 1);
    bugs[bug_index].sendCommand();
    updateBugStatus();
}

private void buttonLeft_Click(object sender, EventArgs e)
{
    int bug_index = int.Parse(comboBugSelect.Text.Substring(4, 1));
    bugs[bug_index].direction = PIVOTLEFT;
    bugs[bug_index].length_index = (byte)(byte.Parse(labelAngleDisp.Text) / 10 - 1);
    bugs[bug_index].sendCommand();
    updateBugStatus();
}

private void buttonRight_Click(object sender, EventArgs e)
{
    int bug_index = int.Parse(comboBugSelect.Text.Substring(4, 1));
    bugs[bug_index].direction = PIVOTRIGHT;
    bugs[bug_index].length_index = (byte)(byte.Parse(labelAngleDisp.Text) / 10 - 1);
    bugs[bug_index].sendCommand();
    updateBugStatus();
}

private void buttonFindPath_Click(object sender, EventArgs e)
{
    if (folderBrowserMATLAB.ShowDialog() == DialogResult.OK) // Some wonderful .Net magic
    {
        textBoxPath.Text = folderBrowserMATLAB.SelectedPath;
    }
}

private void buttonOpen_Click(object sender, EventArgs e)
bool err = false;
_MLAppClass.Execute("closepreview"); //Get matlab to close any old previews

object vid = videoinput('winvideo',1,'RGB24_320x240'); //Open the webcam

try {
    object matlabOut = _MLAppClass.GetVariable("vid", "base");
} catch (System.Runtime.InteropServices.COMException) //Generic error, we'll assume the camera is

wonky {
    matlabDebug.Text += "Eek! Error opening video object... (is the camera plugged in?)\n\n";
    panelCamControls.Enabled = false;
    err = true;
}

if (!err) {
    matlabDebug.Text += "Video object open\n";

    _MLAppClass.Execute("src_vid = getselectedsource(vid);"); //Change things about the camera
    _MLAppClass.Execute("triggerconfig(vid,'manual');"); //So grabbing frames is quick
    _MLAppClass.Execute("cd '" + textBoxPath.Text + '";");
    _MLAppClass.Execute("preview(vid);"); //Open a preview box
    matlabDebug.Text += "Video Preview Started\n";
    panelCamControls.Enabled = true; //Enable the run panel
}

private void matlabDebug_TextChanged(object sender, EventArgs e) {
    matlabDebug.SelectionLength = comPortDebug.Text.Length;
    matlabDebug.ScrollToCaret();
}

private void comPortDebug_TextChanged(object sender, EventArgs e) {
    comPortDebug.SelectionLength = comPortDebug.Text.Length;
    comPortDebug.ScrollToCaret();
}

private void trackBarPan_Scroll(object sender, EventArgs e) {
    _MLAppClass.Execute("set(src_vid, 'Pan', " + trackBarPan.Value + ")");
}

private void trackBarZoom_Scroll(object sender, EventArgs e) {
    _MLAppClass.Execute("set(src_vid, 'Zoom', " + trackBarZoom.Value + ")");
}

private void trackBarTilt_Scroll(object sender, EventArgs e) {
    _MLAppClass.Execute("set(src_vid, 'Tilt', " + trackBarTilt.Value + ")");
}

//Display the calibration windows when necessary
private void trackBarWhiteCal_MouseUp(object sender, MouseEventArgs e) {
    _MLAppClass.Execute("test = getsnapshot(vid)");
    _MLAppClass.Execute("figure(2)");
    _MLAppClass.Execute("imshow(im2bw(test," + trackBarWhiteCal.Value + "/100))");
}

private void trackBarBlackCal_MouseUp(object sender, MouseEventArgs e) {
    _MLAppClass.Execute("test = getsnapshot(vid)");
    _MLAppClass.Execute("figure(3)");
    _MLAppClass.Execute("imshow(im2bw(test," + trackBarBlackCal.Value + "/100))");
}
private void bgWrkerGetCoords_DoWork(object sender, DoWorkEventArgs e) {
    Array XReal = new double[4];
    Array XImag = new double[4];
    int[] CalValues = (int[]).e.Argument;
    double count;
    _MLAppClass.Execute("start(vid);"); //Starts the video object, which should be
    //preset for manual triggering
    for (;;)
    {
        if (bgWrkerGetCoords.CancellationPending)
            break;
        _MLAppClass.Execute("bugs = bugfinder3(getsnapshot(vid)," +
            CalValues[0] + "/100," + //Capture and process and image
            CalValues[1] + "/100, 20");
        _MLAppClass.Execute([m n] = size(bugs);); //Check if there were any results
        try
        {
            count = (double)_MLAppClass.GetVariable("m", "base"); //If (count != 0)  //There were results!
            _MLAppClass.Execute("temp = bugs(1,:);"); //Messy, messy generic types
            _MLAppClass.GetFullMatrix("temp", "base", ref XReal, ref XImag);
            bugs[0].y = (double)XReal.GetValue(0); //Messy, messy generic types
            bugs[0].x = (double)XReal.GetValue(1);
            bugs[0].radius = (double)XReal.GetValue(2);
            bugs[0].angle = (double)XReal.GetValue(3);
            updateBugStatus();
        } catch (System.Runtime.InteropServices.COMException) {
            continue;
        }
        if (count != 0)  //There were results!
        {
            _MLAppClass.Execute("temp = bugs(1,:);");
            _MLAppClass.GetFullMatrix("temp", "base", ref XReal, ref XImag);
            bugs[0].y = (double)XReal.GetValue(0); //Messy, messy generic types
            bugs[0].x = (double)XReal.GetValue(1);
            bugs[0].radius = (double)XReal.GetValue(2);
            bugs[0].angle = (double)XReal.GetValue(3);
            updateBugStatus();
        }
    }
    //Thread.Sleep(0);  //Probably better to include this, but it makes things slow :(
}
private void trackBarXDest_Scroll(object sender, EventArgs e) {
    labelXDestDisp.Text = trackBarXDest.Value.ToString();
}
private void trackBarYDest_Scroll(object sender, EventArgs e) {
    labelYDestDisp.Text = trackBarYDest.Value.ToString();
}
private void panelBugSpace_Paint(object sender, PaintEventArgs e) {
    Point offset = new Point(20,20);  //An offset, to stop us drawing _right_ in the corner
    Graphics _Graphics = e.Graphics;  //Use the form graphics object we were passed
    if(bugs != null)  //If the bugs array has been populated
    {
        foreach (robotBug _robotBug in bugs)
        {
            if (_robotBug.radius != 0) //If this seems to be a real bug
            {
                Rectangle Circle = new Rectangle( //ahahah Rectangle circle. It’s a bounding box
                    Convert.ToInt32(offset.x + _robotBug.x - _robotBug.radius),
                    Convert.ToInt32(offset.y + _robotBug.y - _robotBug.radius),
                    Convert.ToInt32(_robotBug.radius * 2),
                    Convert.ToInt32(_robotBug.radius * 2));
            }
LinearGradientBrush linearBrush =
    new LinearGradientBrush(Color.Red, Color.DarkOrange,
    LinearGradientMode.ForwardDiagonal);
/Graphics.FillEllipse(linearBrush, Circle); //Fill a red ellipse in the circle box

Circle = new Rectangle{
    Convert.ToInt32(offset.X + _robotBug.x - _robotBug.radius * 0.7),
    Convert.ToInt32(offset.Y + _robotBug.y - _robotBug.radius * 0.7),
    Convert.ToInt32(_robotBug.radius * 1.4),
    Convert.ToInt32(_robotBug.radius * 1.4));

SolidBrush solidBrush = new SolidBrush(Color.Black);
/Graphics.FillEllipse(solidBrush, Circle); //Fill a black ellipse over the red one

Pen pen = new Pen(Color.White, Convert.ToInt32(_robotBug.radius * 0.2));
/Graphics.DrawLine(pen, //Draw the white angle line
    Convert.ToInt32(offset.X + _robotBug.x),
    Convert.ToInt32(offset.Y + _robotBug.y),
    Convert.ToInt32(offset.X + _robotBug.x + _robotBug.radius *
    Math.Cos(_robotBug.angle)),
    Convert.ToInt32(offset.Y + _robotBug.y + _robotBug.radius *
    Math.Sin(_robotBug.angle)));

pen = new Pen(Color.Green, 1); //Draw the crosshairs
/Graphics.DrawLine(pen, trackBarXDest.Value + 20, 20, trackBarXDest.Value + 20, 260);
/Graphics.DrawLine(pen, 20, trackBarYDest.Value + 20, 340, trackBarYDest.Value + 20);

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    bgWrkerGetCoords.CancelAsync(); //Try to cleanly exit the background worker
    _MLAppClass.Quit();
}

private void bgWrkerAutoControl_DoWork(object sender, DoWorkEventArgs e)
{
    int[] calc_vals;
    for ( ; ; )
    {
        auto_semaphore.WaitOne();  //Wait for someone to say they're ready for more tasks
        foreach (robotBug _robotbug in bugs)
        {
            if (_robotbug.auto_queue != null
                && _robotbug.ready == true) //Find the bug that's ready
            {
                queue_sem.WaitOne();  //Work out how to get to the next location
                _robotbug.auto_queue[robotbug.auto_stage, 0]  -  Convert.ToInt32(robotbug.x),
                _robotbug.auto_queue[robotbug.auto_stage, 1]  -  Convert.ToInt32(robotbug.y),
                _robotbug.angle); 
                queue_sem.Release();

                if (!_robotbug.move_state) //Turn first
                {
                    if (Math.Abs(calc_vals[1]) >= 16) //If it's a big enough angle to worry
                    {
                        _robotbug.length_index = Math.Abs(calc_vals[1]) / 16 - 1;
                        if (calc_vals[1] < 0)
                            _robotbug.direction = PIVOTLEFT;
                        else
                            _robotbug.direction = PIVOTRIGHT;
                            _robotbug.sendCommand();
                    }
                }
                else
                {
                    if (calc_vals[0] != 0)
                    {
                        _robotbug.length_index = Convert.ToByte((calc_vals[0] * (5 /
                        _robotbug.radius)) / 5 - 1);
                        _robotbug.direction = FORWARD;
                    }
                }
            }
        }
    }
}
Find the angle the destination is

private
private
private
private
queue_sem
int
catch
try
queue_sem
bugs
bugs
bugs
bugs
bugs
bugs
bugs
queue_sem
int
return
if
if
calc_vals
else
calc_vals
return
if
if
calc_vals
int
(y == 0 || y == 0)
calc_vals[1] = 0; //We need to go in a straigh line somewhere
else
/*Find the angle*/
if (x <= 0) //Find the quadrant
if (y <= 0)
calc_vals[1] = -180 + (calc_vals[1]);
else
calc_vals[1] = 180 - (calc_vals[1]);
else
if (y < 0)
calc_vals[1] = -(calc_vals[1]);
calc_vals[1] = Convert.ToInt32(180*Math.PI);
if (calc_vals[1] > 180)
calc_vals[1] -= 360; //Turning a long way left is the same as turning a little bit right :D
if (calc_vals[1] < -180)
calc_vals[1] += 360;
return calc_vals;
}

private void buttonAutoGo_Click(object sender, EventArgs e)
{
int bug_index = int.Parse(comboAutoBugSelect.Text.Substring(4, 1));
queue_sem.WaitOne();
bugs[bug_index].auto_queue = new int[1, 2]; //Add an item to the queue
bugs[bug_index].auto_queue[0, 0] = trackBarXDest.Value;
bugs[bug_index].auto_queue[0, 1] = trackBarYDest.Value;
bugs[bug_index].repeat = false;
bugs[bug_index].auto_stage = 0;
bugs[bug_index].move_state = false;
bugs[bug_index].ready = true;
queue_sem.Release();
try { auto_semaphore.Release(); } //Tell the queuing system we made a change
catch(SemaphoreFullException) { };}

private void buttonSquare_Click(object sender, EventArgs e)
{
int bug_index = int.Parse(comboAutoBugSelect.Text.Substring(4, 1));
queue_sem.WaitOne();
bugs[bug_index].auto_queue = new int[2, 2]; //Add a lot of points to the queue
bugs[bug_index].auto_queue[0, 0] = 100;
bugs[bug_index].auto_queue[0, 1] = 100;
bugs[bug_index].auto_queue[1, 0] = 200;
bugs[bug_index].auto_queue[1, 1] = 200;

bugs[bug_index].repeat = true;
bugs[bug_index].ready = true;
bugs[bug_index].auto_stage = 0;
bugs[bug_index].move_state = false;
queue_sem.Release();

try { auto_semaphore.Release(); } //Tell the queue we changes something
catch (SemaphoreFullException) { }
}

public class robotBug
{
    public Form1 _Form;
    public bool ready;
    public SerialPort _serialPort;
    public Semaphore acksem;

    public byte sequence;
    public byte srcaddress;
    public byte dstaddress;
    public byte direction;
    public byte length_index;
    public byte[] packet;

    public int auto_stage;
    public bool move_state;
    public int[,] auto_queue;
    public bool repeat;

    public double x;
    public double y;
    public double angle;
    public double radius;

    public void sendCommand() //Complete sending operation, including auto retrys
    {
        Random rand = new Random();
        bool _continue = true;
        int repeat = 0;
        packet[0] = 0xAA;
        packet[1] = dstaddress;
        packet[2] = srcaddress;
        packet[3] = direction;
        packet[4] = length_index;
        packet[5] = sequence++;
        packet[6] = CRC8();

        while (_continue)
        {
            /*for (int i = 0; i < 5; i++)
            {
                _serialPort.Write(packet, 0, 1);
            }*/
            _serialPort.Write(packet, 0, packet.Length);
            _continue = !acksem.WaitOne(rand.Next(50, 150), false);
            ready = _continue;
            if (++repeat > 100)
            {
                _Form.writeRunDebug("Timed out sending to " + dstaddress + " :(
                _Form.textRunDebug.AppendText("Timed out sending to " + dstaddress + " :(
                _continue = false;
            }
        }
    }
public void sendCommandNoRepeat() //Used for one off sends (like an ACK)
{
    packet[0] = 0xAA;
    packet[1] = dstaddress;
    packet[2] = srcaddress;
    packet[3] = direction;
    packet[4] = length_index;
    packet[5] = sequence++;
    packet[6] = CRC8();

    _serialPort.Write(packet, 0, packet.Length);
}

public byte CRC8()
{
    byte CRC = 0xFF;
    for (int i = 1; i < (packet.Length - 1); i++)
    {
        CRC = (byte)(CRC ^ packet[i]);
        for (int j = 0; j < 8; j++)
            CRC = ((CRC & 0x80) != 0) ? (byte)((CRC << 1) ^ 0x1D) : (byte)(CRC << 1);
    }
    return CRC;
}

public void recvPacket() //Just populate the members from the packet
{
    dstaddress = packet[1];
    srcaddress = packet[2];
    direction = packet[3];
    length_index = packet[4];
    sequence = packet[5];
}

}
this.groupBox2 = new System.Windows.Forms.GroupBox();
this.panelCamControls = new System.Windows.Forms.Panel();
this.label18 = new System.Windows.Forms.Label();
this.label19 = new System.Windows.Forms.Label();
this.trackBarBlackCal = new System.Windows.Forms.TrackBar();
this.trackBarWhiteCal = new System.Windows.Forms.TrackBar();
this.label18 = new System.Windows.Forms.Label();
this.trackBarZoom = new System.Windows.Forms.TrackBar();
this.label17 = new System.Windows.Forms.Label();
this.trackBarTilt = new System.Windows.Forms.TrackBar();
this.label16 = new System.Windows.Forms.Label();
this.trackBarPan = new System.Windows.Forms.TrackBar();
this matlabDebug = new System.Windows.Forms.TextBox();
this.buttonOpen = new System.Windows.Forms.Button();
this.buttonFindPath = new System.Windows.Forms.Button();
this.label15 = new System.Windows.Forms.Label();
this.textBoxPath = new System.Windows.Forms.TextBox();
this.groupBox1 = new System.Windows.Forms.GroupBox();
this.buttonComConnect = new System.Windows.Forms.Button();
this.comPortDebug = new System.Windows.Forms.TextBox();
this.labelBaud = new System.Windows.Forms.Label();
this.listBoxBauds = new System.Windows.Forms.ListBox();
this.labelComHeading = new System.Windows.Forms.Label();
this.listBoxPorts = new System.Windows.Forms.ListBox();
this.tabRun = new System.Windows.Forms.TabPage();
this.panelRun = new System.Windows.Forms.Panel();
this.panelBugSpace = new System.Windows.Forms.Panel();
this.groupBox3 = new System.Windows.Forms.GroupBox();
this.buttonSquare = new System.Windows.Forms.Button();
this.buttonAutoGo = new System.Windows.Forms.Button();
this.labelYDestDisp = new System.Windows.Forms.Label();
this.label14 = new System.Windows.Forms.Label();
this.trackBarYDest = new System.Windows.Forms.TrackBar();
this.labelXDestDisp = new System.Windows.Forms.Label();
this.label12 = new System.Windows.Forms.Label();
this.trackBarXDest = new System.Windows.Forms.TrackBar();
this.comboBoxAutoBugSelect = new System.Windows.Forms.ComboBox();
this.textRunDebug = new System.Windows.Forms.TextBox();
this.groupBoxManual = new System.Windows.Forms.GroupBox();
this.linkLabelAngleDisp = new System.Windows.Forms.Label();
this.labelDistanceDisp = new System.Windows.Forms.Label();
this.label14 = new System.Windows.Forms.Label();
this.label13 = new System.Windows.Forms.Label();
this.trackBarDistance = new System.Windows.Forms.TrackBar();
this.trackBarAngle = new System.Windows.Forms.TrackBar();
this.comboBoxBugSelect = new System.Windows.Forms.ComboBox();
this.buttonRight = new System.Windows.Forms.Button();
this.buttonLeft = new System.Windows.Forms.Button();
this.buttonForward = new System.Windows.Forms.Button();
this.buttonSetBugs = new System.Windows.Forms.Button();
this.label12 = new System.Windows.Forms.Label();
this.textBoxBugStatus = new System.Windows.Forms.TextBox();
this.numBugs = new System.Windows.Forms.NumericUpDown();
this.folderBrowserMATLAB = new System.Windows.Forms.FolderBrowserDialog();
this.bgWorkerGetCoords = new System.ComponentModel.BackgroundWorker();
this.bgWorkerAutoControl = new System.ComponentModel.BackgroundWorker();
this.tabMain.SuspendLayout();
this.tabConfig.SuspendLayout();
this.groupBox2.SuspendLayout();
this.panelCamControls.SuspendLayout();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarBlackCal)).BeginInit();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarWhiteCal)).BeginInit();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarZoom)).BeginInit();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarTilt)).BeginInit();
((System.ComponentModel.IsSupportInitialize) ((this.trackBarPan)).BeginInit();
this.groupBox1.SuspendLayout();
this.tabRun.SuspendLayout();
this.panelRun.SuspendLayout();
this.groupBox3.SuspendLayout();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarYDest)).BeginInit();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarXDest)).BeginInit();
this.groupBox1.SuspendLayout();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarDistance)).BeginInit();
(System.ComponentModel.IsSupportInitialize) ((this.trackBarAngle)).BeginInit();
((System.ComponentModel.ISupportInitialize)$this).BeginInit();
this.SuspendLayout();

//tabMain
this.tabMain.Controls.Add(this.tabConfig);
this.tabMain.Controls.Add(this.tabRun);
this.tabMain.Location = new System.Drawing.Point(12, 12);
this.tabMain.Name = "tabMain";
this.tabMain.SelectedIndex = 0;
this.tabMain.Size = new System.Drawing.Size(813, 468);
this.tabMain.TabIndex = 0;

//tabConfig
this.tabConfig.Controls.Add(this.label1);
this.tabConfig.Controls.Add(this.groupBox2);
this.tabConfig.Controls.Add(this.groupBox1);
this.tabConfig.Location = new System.Drawing.Point(4, 22);
this.tabConfig.Name = "tabConfig";
this.tabConfig.Size = new System.Drawing.Size(805, 434);
this.tabConfig.TabIndex = 0;
this.tabConfig.Text = "Config";
this.tabConfig.UseVisualStyleBackColor = true;

//label1
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(649, 175);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(135, 52);
this.label1.TabIndex = 2;
this.label1.Text = "eBug Control Panel\n\nDavid McKechnie\nDept ECSE\nMonash University (2008)";

//groupBox2
this.groupBox2.Controls.Add(this.panelCamControls);
this.groupBox2.Controls.Add(this.matlabDebug);
this.groupBox2.Controls.Add(this.buttonOpen);
this.groupBox2.Controls.Add(this.buttonFindPath);
this.groupBox2.Controls.Add(this.label15);
this.groupBox2.Controls.Add(this(textBoxPath);
this.groupBox2.Location = new System.Drawing.Point(336, 30);
this.groupBox2.Name = "groupBox2";
this.groupBox2.Size = new System.Drawing.Size(307, 370);
this.groupBox2.TabIndex = 1;
this.groupBox2.TabStop = false;
this.groupBox2.Text = "Camera Interface";

//panelCamControls
this.panelCamControls.Controls.Add(this.label10);
this.panelCamControls.Controls.Add(this.label9);
this.panelCamControls.Controls.Add(this.trackBarBlackCal);
this.panelCamControls.Controls.Add(this.trackBarWhiteCal);
this.panelCamControls.Controls.Add(this.label8);
this.panelCamControls.Controls.Add(this.trackBarZoom);
this.panelCamControls.Controls.Add(this.label17);
this.panelCamControls.Controls.Add(this.trackBarTilt);
this.panelCamControls.Controls.Add(this.label16);
this.panelCamControls.Controls.Add(this.trackBarPan);
this.panelCamControls.Enabled = false;
this.panelCamControls.Location = new System.Drawing.Point(9, 102);
this.panelCamControls.Name = "panelCamControls";
this.panelCamControls.Size = new System.Drawing.Size(307, 155);
this.panelCamControls.TabIndex = 7;

//label10

this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(150, 105);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(50, 13);
this.label10.TabIndex = 14;
this.label10.Text = "BlackCal";
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(146, 55);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(54, 13);
this.label9.TabIndex = 13;
this.label9.Text = "WhiteCal";
//
// trackBarBlackCal
//
this.trackBarBlackCal.Location = new System.Drawing.Point(197, 105);
this.trackBarBlackCal.Maximum = 100;
this.trackBarBlackCal.Name = "trackBarBlackCal";
this.trackBarBlackCal.Size = new System.Drawing.Size(104, 45);
this.trackBarBlackCal.TabIndex = 12;
this.trackBarBlackCal.TickFrequency = 10;
this.trackBarBlackCal.Value = 90;
this.trackBarBlackCal.MouseUp += new System.Windows.Forms.MouseEventHandler(this.trackBarBlackCal_MouseUp);
//
// trackBarWhiteCal
//
this.trackBarWhiteCal.Location = new System.Drawing.Point(197, 54);
this.trackBarWhiteCal.Maximum = 100;
this.trackBarWhiteCal.Name = "trackBarWhiteCal";
this.trackBarWhiteCal.Size = new System.Drawing.Size(104, 45);
this.trackBarWhiteCal.TabIndex = 7;
this.trackBarWhiteCal.TickFrequency = 10;
this.trackBarWhiteCal.Value = 50;
this.trackBarWhiteCal.MouseUp += new System.Windows.Forms.MouseEventHandler(this.trackBarWhiteCal_MouseUp);
//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(97, 105);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(46, 13);
this.label8.TabIndex = 11;
this.label8.Text = "Zoom";
//
// trackBarZoom
//
this.trackBarZoom.Location = new System.Drawing.Point(3, 105);
this.trackBarZoom.Maximum = 200;
this.trackBarZoom.Name = "trackBarZoom";
this.trackBarZoom.Size = new System.Drawing.Size(104, 45);
this.trackBarZoom.TabIndex = 10;
this.trackBarZoom.TickFrequency = 40;
this.trackBarZoom.Scroll += new System.EventHandler(this.trackBarZoom_Scroll);
//
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(101, 55);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(22, 13);
this.label7.TabIndex = 9;
this.label7.Text = "Tilt";
//
// trackBarTilt
//
this.trackBarTilt.Location = new System.Drawing.Point(3, 54);
this.trackBarTilt.Maximum = 180;
this.trackBarTilt.Minimum = -180;
this.trackBarTilt.Name = "trackBarTilt";
this.trackBarTilt.Size = new System.Drawing.Size(184, 45);
this.trackBarTilt.TabIndex = 8;
this.trackBarTilt.TickFrequency = 60;
this.trackBarTilt.Scroll += new System.EventHandler(this.trackBarTilt_Scroll);

// label6
//
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(97, 3);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(26, 13);
this.label6.TabIndex = 7;
this.label6.Text = "Pan";

// trackBarPan
//
this.trackBarPan.Location = new System.Drawing.Point(3, 3);
this.trackBarPan.Maximum = 180;
this.trackBarPan.Minimun = -180;
this.trackBarPan.Name = "trackBarPan";
this.trackBarPan.Size = new System.Drawing.Size(104, 45);
this.trackBarPan.TabIndex = 6;
this.trackBarPan.TickFrequency = 60;
this.trackBarPan.Scroll += new System.EventHandler(this.trackBarPan_Scroll);

// matlabDebug
//
this.matlabDebug.AcceptsReturn = true;
this.matlabDebug.Location = new System.Drawing.Point(6, 263);
this.matlabDebug.Multiline = true;
this.matlabDebug.Name = "matlabDebug";
this.matlabDebug.ReadOnly = true;
this.matlabDebug.Size = new System.Drawing.Size(299, 101);
this.matlabDebug.TabIndex = 4;
this.matlabDebug.TextChanged += new System.EventHandler(this.matlabDebug_TextChanged);

// buttonOpen
//
this.buttonOpen.Location = new System.Drawing.Point(158, 60);
this.buttonOpen.Name = "buttonOpen";
this.buttonOpen.Size = new System.Drawing.Size(116, 36);
this.buttonOpen.TabIndex = 5;
this.buttonOpen.Text = "Start Video Preview";
this.buttonOpen.UseSystemSelectorsColor = true;
this.buttonOpen.Click += new System.EventHandler(this.buttonOpen_Click);

// buttonFindPath
//
this.buttonFindPath.Location = new System.Drawing.Point(246, 33);
this.buttonFindPath.Name = "buttonFindPath";
this.buttonFindPath.Size = new System.Drawing.Size(28, 21);
this.buttonFindPath.TabIndex = 4;
this.buttonFindPath.Text = "...";
this.buttonFindPath.UseSystemSelectorsColor = true;
this.buttonFindPath.Click += new System.EventHandler(this.buttonFindPath_Click);

// label5
//
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(6, 18);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(117, 13);
this.label5.TabIndex = 3;
this.label5.Text = "Path to bugfinder3.m";

// textBoxPath
//
this.textBoxPath.Location = new System.Drawing.Point(6, 32);
```csharp
this.textBoxPath.Name = "textBoxPath";
this.textBoxPath.Size = new System.Drawing.Size(234, 22);
this.textBoxPath.TabIndex = 0;
this.textBoxPath.Text = "C:\Users\Public\Final year project";
//
// groupBox1
//
this groupBox1.Controls.Add(this.buttonComConnect);
this groupBox1.Controls.Add(this.comPortDebug);
this groupBox1.Controls.Add(this.labelBaud);
this groupBox1.Controls.Add(this.listBoxBauds);
this groupBox1.Controls.Add(this.labelBauds);
this groupBox1.Controls.Add(this.listBoxComPorts);
this groupBox1.Location = new System.Drawing.Point(30, 30);
this groupBox1.Name = "groupBox1";
this groupBox1.Size = new System.Drawing.Size(280, 376);
this groupBox1.TabIndex = 8;
this groupBox1.TabStop = false;
this groupBox1.Text = "Serial Interface";
//
// buttonComConnect
//
this buttonComConnect.Font = new System.Drawing.Font("Segoe UI", 8.25F,
this buttonComConnect.Location = new System.Drawing.Point(100, 130);
this buttonComConnect.Name = "buttonComConnect";
this buttonComConnect.Size = new System.Drawing.Size(75, 48);
this buttonComConnect.TabIndex = 4;
this buttonComConnect.Text = "Connect!";
this buttonComConnect.UseVisualStyleBackColor = true;
this buttonComConnect.Click += new System.EventHandler(this.buttonComConnect_Click);
//
// comPortDebug
//
this comPortDebug.AutoSize = true;
this comPortDebug.Location = new System.Drawing.Point(6, 207);
this comPortDebug.Multiline = true;
this comPortDebug.Name = "comPortDebug";
this comPortDebug.ReadOnly = true;
this comPortDebug.Size = new System.Drawing.Size(267, 157);
this comPortDebug.TabIndex = 3;
this comPortDebug.TextChanged += new System.EventHandler(this.comPortDebug_TextChanged);
//
// labelBaud
//
this labelBaud.AutoSize = true;
this labelBaud.Location = new System.Drawing.Point(215, 16);
this labelBaud.Name = "labelBaud";
this labelBaud.Size = new System.Drawing.Size(68, 13);
this labelBaud.TabIndex = 2;
this labelBaud.Text = "Baud Rate";
//
// listBoxBauds
//
this listBoxBauds.Font = new System.Drawing.Font("Segoe UI", 9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this listBoxBauds.FormattingEnabled = true;
this listBoxBauds.ItemHeight = 17;
this listBoxBauds.Items.AddRange(new object[] { "1200", "14400", "19200", "9600" });
this listBoxBauds.Location = new System.Drawing.Point(153, 32);
this listBoxBauds.Name = "listBauds";
this listBoxBauds.Size = new System.Drawing.Size(120, 89);
this listBoxBauds.SortedList = true;
this listBoxBauds.TabIndex = 2;
//
// labelLComHeading
//
this labelLComHeading.AutoSize = true;
this labelLComHeading.Location = new System.Drawing.Point(76, 16);
```

this.labelComHeading.Name = "labelComHeading";
this.labelComHeading.Size = new System.Drawing.Size(54, 13);
this.labelComHeading.TabStop = true;
this.labelComHeading.Text = "Com Port";

// listComPorts
this.listComPorts.FormattingEnabled = true;
this.listComPorts.ItemHeight = 17;
this.listComPorts.Location = new System.Drawing.Point(6, 32);
this.listComPorts.Name = "listComPorts";
this.listComPorts.Size = new System.Drawing.Size(128, 89);
this.listComPorts.TabIndex = 8;

// tabPage3
this.tabPage3.Controls.Add(this.panelBugSpace);
this.tabPage3.Controls.Add(this.groupBox3);
this.tabPage3.Controls.Add(this.textBoxBugStatus);
this.tabPage3.Controls.Add(this.numBugs);
this.tabPage3.Controls.Add(this.buttonSetBugs);
this.tabPage3.Controls.Add(this.label2);
this.tabPage3.Controls.Add(this.tabPage2);
this.tabPage3.Location = new System.Drawing.Point(360, 0);
this.tabPage3.Name = "tabPage3";
this.tabPage3.Size = new System.Drawing.Size(89, 434);
this.tabPage3.TabIndex = 5;
this.tabPage3.Paint += new System.Windows.Forms.PaintEventHandler(this.tabPage3_Paint);

// panelBugSpace
this.panelBugSpace.Location = new System.Drawing.Point(421, 79);
this.panelBugSpace.Name = "panelBugSpace";
this.panelBugSpace.Size = new System.Drawing.Size(360, 288);
this.panelBugSpace.TabIndex = 0;

// groupBox3
this.groupBox3.Controls.Add(this.buttonSquare);
this.groupBox3.Controls.Add(this.buttonAutoGo);
this.groupBox3.Controls.Add(this.labelYDestDisp);
this.groupBox3.Controls.Add(this.label14);
this.groupBox3.Controls.Add(this.trackBarYDest);
this.groupBox3.Controls.Add(this.label12);
this.groupBox3.Controls.Add(this.label11);
this.groupBox3.Controls.Add(this.trackBarXDest);
this.groupBox3.Controls.Add(this.comboBoxAutoBugSelect);
this.groupBox3.Location = new System.Drawing.Point(6, 162);
this.groupBox3.Name = "groupBox3";
this.groupBox3.Size = new System.Drawing.Size(283, 180);
this.groupBox3.TabIndex = 7;
this.groupBox3.TabStop = false;
this.groupBox3.Text = "Automatic Control";

// buttonSquare
this.buttonSquare.Location = new System.Drawing.Point(10, 124);
this.buttonSquare.Name = "buttonSquare";
this.buttonSquare.Size = new System.Drawing.Size(75, 42);
this.buttonSquare.TabIndex = 16;
this.buttonSquare.Text = "Square";
this.buttonSquare.UseVisualStyleBackColor = true;
this.buttonSquare.Click += new System.EventHandler(this.buttonSquare_Click);

// buttonAutoGo

this.buttonAutoGo.Location = new System.Drawing.Point(150, 54);
this.buttonAutoGo.Name = "buttonAutoGo";
this.buttonAutoGo.Size = new System.Drawing.Size(47, 33);
this.buttonAutoGo.TabIndex = 15;
this.buttonAutoGo.Text = "Go";
this.buttonAutoGo.UseVisualStyleBackColor = true;
this.buttonAutoGo.Click += new System.EventHandler(this.buttonAutoGo_Click);

// labelYDestDisp

this.labelYDestDisp.AutoSize = true;
this.labelYDestDisp.Location = new System.Drawing.Point(18, 86);
this.labelYDestDisp.Name = "labelYDestDisp";
this.labelYDestDisp.Size = new System.Drawing.Size(25, 13);
this.labelYDestDisp.TabIndex = 14;
this.labelYDestDisp.Text = "y Dest";
this.labelYDestDisp.TextAlign = System.Drawing.ContentAlignmentMiddleRight;

// label14

this.label14.AutoSize = true;
this.label14.Location = new System.Drawing.Point(5, 74);
this.label14.Name = "label14";
this.label14.Size = new System.Drawing.Size(38, 13);
this.label14.TabIndex = 13;
this.label14.Text = "y Dest";
this.label14.TextAlign = System.Drawing.ContentAlignmentMiddleRight;

// trackBarYDest

this.trackBarYDest.Location = new System.Drawing.Point(38, 73);
this.trackBarYDest.Maximum = 240;
this.trackBarYDest.Name = "trackBarYDest";
this.trackBarYDest.Size = new System.Drawing.Size(116, 45);
this.trackBarYDest.SmallChange = 10;
this.trackBarYDest.TabIndex = 12;
this.trackBarYDest.TickFrequency = 10;
this.trackBarYDest.Value = 120;
this.trackBarYDest.Scroll += new System.EventHandler(this.trackBarYDest_Scroll);

// labelXDestDisp

this.labelXDestDisp.AutoSize = true;
this.labelXDestDisp.Location = new System.Drawing.Point(18, 54);
this.labelXDestDisp.Name = "labelXDestDisp";
this.labelXDestDisp.Size = new System.Drawing.Size(25, 13);
this.labelXDestDisp.TabIndex = 13;
this.labelXDestDisp.Text = "x Dest";
this.labelXDestDisp.TextAlign = System.Drawing.ContentAlignmentMiddleRight;

// label12

this.label12.AutoSize = true;
this.label12.Location = new System.Drawing.Point(5, 42);
this.label12.Name = "label12";
this.label12.Size = new System.Drawing.Size(38, 13);
this.label12.TabIndex = 10;
this.label12.Text = "x Dest";
this.label12.TextAlign = System.Drawing.ContentAlignmentMiddleRight;

// trackBarXDest

this.trackBarXDest.BackColor = System.Drawing.SystemColors.Window;
this.trackBarXDest.Location = new System.Drawing.Point(38, 42);
this.trackBarXDest.Maximum = 328;
this.trackBarXDest.Name = "trackBarXDest";
this.trackBarXDest.Size = new System.Drawing.Size(116, 45);
this.trackBarXDest.SmallChange = 10;
this.trackBarXDest.TabIndex = 8;
this.trackBarXDest.TickFrequency = 10;
this.trackBarXDest.Value = 160;
this.trackBarXDest.Scroll += new System.EventHandler(this.trackBarXDest_Scroll);

// comboAutoBugSelect
this.comboAutoBugSelect.FormattingEnabled = true;
this.comboAutoBugSelect.Location = new System.Drawing.Point(6, 15);
this.comboAutoBugSelect.Name = "comboAutoBugSelect";
this.comboAutoBugSelect.Size = new System.Drawing.Size(121, 21);
this.comboAutoBugSelect.TabIndex = 2;

// textRunDebug
this.textRunDebug.Location = new System.Drawing.Point(6, 348);
this.textRunDebug.Multiline = true;
this.textRunDebug.Name = "textRunDebug";
this.textRunDebug.ReadOnly = true;
this.textRunDebug.Size = new System.Drawing.Size(377, 83);
this.textRunDebug.TabIndex = 6;

// groupManual
this.groupManual.Controls.Add(this.labelAngleDisp);
this.groupManual.Controls.Add(this.labelDistanceDisp);
this.groupManual.Controls.Add(this.label14);
this.groupManual.Controls.Add(this.label3);
this.groupManual.Controls.Add(this.trackBarDistance);
this.groupManual.Controls.Add(this.trackBarAngle);
this.groupManual.Controls.Add(this.comboBugSelect);
this.groupManual.Controls.Add(this.buttonRight);
this.groupManual.Controls.Add(this.buttonLeft);
this.groupManual.Controls.Add(this.buttonForward);
this.groupManual.Enabled = false;
this.groupManual.Location = new System.Drawing.Point(215, 162);
this.groupManual.Name = "groupManual";
this.groupManual.Size = new System.Drawing.Size(168, 180);
this.groupManual.TabIndex = 3;
this.groupManual.TabStop = false;
this.groupManual.Text = "Manual Control";

// labelAngleDisp
this.labelXDisp.AutoSize = true;
this.labelXDisp.Location = new System.Drawing.Point(25, 86);
this.labelXDisp.Name = "labelAngleDisp";
this.labelXDisp.Size = new System.Drawing.Size(19, 13);
this.labelXDisp.TabIndex = 9;
this.labelXDisp.Text = "10";
this.labelXDisp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

// labelDistanceDisp
this.labelXDisp.AutoSize = true;
this.labelXDisp.Location = new System.Drawing.Point(32, 137);
this.labelXDisp.Name = "labelDistanceDisp";
this.labelXDisp.Size = new System.Drawing.Size(13, 13);
this.labelXDisp.TabIndex = 6;
this.labelXDisp.Text = "5";
this.labelXDisp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

// label4
this.labelXDisp.AutoSize = true;
this.labelXDisp.Location = new System.Drawing.Point(0, 124);
this.labelXDisp.Name = "label4";
this.label4.Size = new System.Drawing.Size(50, 13);
this.label4.TabIndex = 8;
this.label4.Text = "Dist (cm)"
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(9, 73);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(37, 13);
this.label3.TabIndex = 7;
this.label3.Text = "Angle";
//
// trackBarDistance
//
this.trackBarDistance.BackColor = System.Drawing.SystemColors.Window;
this.trackBarDistance.Location = new System.Drawing.Point(58, 124);
this.trackBarDistance.Maximum = 150;
this.trackBarDistance.Minimum = 5;
this.trackBarDistance.Name = "trackBarDistance";
this.trackBarDistance.Size = new System.Drawing.Size(104, 45);
this.trackBarDistance.TabIndex = 6;
this.trackBarDistance.TickFrequency = 10;
this.trackBarDistance.Value = 5;
this.trackBarDistance.Scroll += new System.EventHandler(this.trackBarDistance_Scroll);
//
// trackBarAngle
//
this.trackBarAngle.Location = new System.Drawing.Point(58, 73);
this.trackBarAngle.Maximum = 180;
this.trackBarAngle.Minimum = 10;
this.trackBarAngle.Name = "trackBarAngle";
this.trackBarAngle.Size = new System.Drawing.Size(104, 45);
this.trackBarAngle.SmallChange = 10;
this.trackBarAngle.TabIndex = 6;
this.trackBarAngle.TickFrequency = 10;
this.trackBarAngle.Value = 10;
this.trackBarAngle.Scroll += new System.EventHandler(this.trackBarAngle_Scroll);
//
// comboBugSelect
//
this.comboBugSelect_FormattingEnabled = true;
this.comboBugSelect.Location = new System.Drawing.Point(6, 15);
this.comboBugSelect.Name = "comboBugSelect";
this.comboBugSelect.Size = new System.Drawing.Size(121, 21);
this.comboBugSelect.TabIndex = 1;
//
// buttonRight
//
this.buttonRight.Location = new System.Drawing.Point(87, 42);
this.buttonRight.Name = "buttonRight";
this.buttonRight.Size = new System.Drawing.Size(40, 25);
this.buttonRight.TabIndex = 0;
this.buttonRight.Text = "Rght";
this.buttonRight.UseVisualStyleBackColor = true;
this.buttonRight.Click += new System.EventHandler(this.buttonRight_Click);
//
// buttonLeft
//
this.buttonLeft.Location = new System.Drawing.Point(6, 42);
this.buttonLeft.Name = "buttonLeft";
this.buttonLeft.Size = new System.Drawing.Size(40, 25);
this.buttonLeft.TabIndex = 0;
this.buttonLeft.Text = "Left";
this.buttonLeft.UseVisualStyleBackColor = true;
this.buttonLeft.Click += new System.EventHandler(this.buttonLeft_Click);
//
// buttonForward
//
this.buttonForward.Location = new System.Drawing.Point(46, 42);
this.buttonForward.Name = "buttonForward";
this.buttonForward.Size = new System.Drawing.Size(40, 25);
this.buttonForward.TabIndex = 0;
this.buttonForward.Text = "Fwd";
this.buttonForward.UseVisualStyleBackColor = true;
this.buttonForward.Click += new System.EventHandler(this.buttonForward_Click);

// buttonSetBugs
this.buttonSetBugs.Location = new System.Drawing.Point(97, 3);
this.buttonSetBugs.Name = "buttonSetBugs";
this.buttonSetBugs.Size = new System.Drawing.Size(78, 22);
this.buttonSetBugs.TabIndex = 4;
this.buttonSetBugs.UseVisualStyleBackColor = true;
this.buttonSetBugs.Click += new System.EventHandler(this.buttonSetBugs_Click);

// label2
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(5, 5);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(46, 13);
this.label2.TabIndex = 3;
this.label2.Text = "# Bugs:");

// textboxBugStatus
this.textboxBugStatus.Location = new System.Drawing.Point(5, 25);
this.textboxBugStatus.Multiline = true;
this.textboxBugStatus.Name = "textboxBugStatus";
this.textboxBugStatus.ReadOnly = true;
this.textboxBugStatus.Size = new System.Drawing.Size(380, 127);
this.textboxBugStatus.TabIndex = 2;

// numBugs
this.numBugs.Location = new System.Drawing.Point(53, 3);
this.numBugs.Maximum = new decimal(new int[] { 8, 0, 0, 0 });
this.numBugs.Minimum = new decimal(new int[] { 1, 0, 0, 0 });
this.numBugs.Name = "numBugs";
this.numBugs.Size = new System.Drawing.Size(38, 22);
this.numBugs.TabIndex = 1;
this.numBugs.Value = new decimal(new int[] { 1, 0, 0, 0 });

// bgWrkerGetCoords
this.bgWrkerGetCoords.WorkerSupportsCancellation = true;
this.bgWrkerGetCoords.DoWork += new System.ComponentModel.DoWorkEventHandler(this.bgWrkerGetCoords_DoWork);

// bgWrkerAutoControl
this.bgWrkerAutoControl.WorkerSupportsCancellation = true;
this.bgWrkerAutoControl.DoWork += new System.ComponentModel.DoWorkEventHandler(this.bgWrkerAutoControl_DoWork);

// Form1
this.AutoScaleBaseSize = new System.Drawing.Size(6F, 13F);
this.ClientSize = new System.Drawing.Size(834, 484);
this.Controls.Add(this.tabMain);
this.MaximizeBox = false;
this.MaximizeBox = new System.Drawing.Size(580, 520);
this.MaximizeBox = new System.Drawing.Size(580, 520);
this.Name = "Form1";
this.Text = "Bug Controls";
this.Load += new System.EventHandler(this.Form1_Load);
this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form1_FormClosing);
this.tabControl1Collapsed = false;
this.tabControl1Collapsed = false;
this.tabControl1Collapsed = false;
this.tabControl1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Collapsed = false;
this.tabPage1Colla...
iii. Program.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace eBugControl
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        /// <STAThread>
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

B. Matlab Source Code

i. bugfinder3.m

```matlab
function circles = bugfinder3(img_in,blackthres,whitethres,gridsize)

circles = [];
img_black = im2bw(img_in, blackthres);
img_white = im2bw(img_in, whitethres);
grid1 = 1:gridsize:size(img_in,1);
grid2 = 1:gridsize:size(img_in,2);
img_black_small = img_black(grid1,grid2);
index = 1;
```
function limits = limitfinder(img_white, i, j, gridsize)

i_big = (i-1)*gridsize+1;
j_big = (j-1)*gridsize+1;
n = i_big;
detectstate = 0;
umblack = 0;
fail = 0;

%First find the bottom of the circle
while(numblack < 1)
    if(img_white(n, j_big))
        % Found another white pixel
        if(detectstate)
            numblack = 0;
detectstate = 0;
        end
    else
        %Found a black pixel
        detectstate = 1;
umblack = numblack + 1;
    end
    n = n+1;
    if((abs(n-i_big)>60)||(n>size(img_white,1)))
fail = 1;
break;
end
bottom = n;

%Then find the top of the circle
n = i_big;
detectstate = 0;
umblack = 0;

while(numblack < 1)
    if(img_white(n, j_big))
        % Found another white pixel
        if(detectstate)
            numblack = 0;
detectstate = 0;
        end
    else
        %Found a black pixel
        detectstate = 1;
umblack = numblack + 1;
    end
    n = n-1;
    if((abs(n-i_big)>60)||(n<1))
fail = 1;
break;
end

top = n;

%Find the right limit
n = j_big;
detectstate = 0;
umblack = 0;

while(numblack < 1)
    if(img_white(i_big,n))
        % Found another white pixel
        if(detectstate)
            numblack = 0;
            detectstate = 0;
        end
    else
        % Found a black pixel
        detectstate = 1;
        numblack = numblack + 1;
    end
    n = n+1;
    if(abs(n-j_big)>60)||(n>size(img_white,2))
        fail = 1;
        break;
    end
end
right = n;
if(~fail)
    M = [top^2+j_big^2, top, j_big, 1;
        bottom^2-j_big^2, bottom, j_big, 1;
        i_big^2+right^2, i_big, right, 1];
    m_11 = det(M(:,[2 3 4]));
    m_12 = det(M(:,[1 3 4]));
    m_13 = det(M(:,[1 2 4]));
    m_14 = det(M(:,[1 2 3]));
    y = 0.5*m_12/m_11;
    x = -0.5*m_13/m_11;
    r = sqrt(y^2 + x^2 + m_14/m_11);
    limits = [y x r 0];
else
    limits = [0 0 0 1];
end
end

function angle = findangle(y0, x0, r, img_black)
    r = round(0.6*r);
    x0 = round(x0);
    y0 = round(y0);
    angle = [];
if((x0-r>0)&&(x0+r<size(img_black,2))&&(y0-r>0)&&(y0+r<size(img_black,1)))
    n = 1;
    f = 1-r;
    ddF_x = 0;
    ddF_y = 0;
    x = 0;
    y = r;
    if(~img_black(y0+r,x0))
        blackpix(n,:) = [y0+r, x0];
        n = n + 1;
    end
    if(~img_black(y0-r,x0))
        blackpix(n,:) = [y0-r, x0];
        n = n + 1;
    end
    if(~img_black(y0,x0+r))
        blackpix(n,:) = [y0, x0+r];
        n = n + 1;
    end
    if(~img_black(y0,x0-r))
        blackpix(n,:) = [y0, x0-r];
        n = n + 1;
end
while (x < y)
    if (f >= 0)
        y = y - 1;
        ddf_y = ddf_y + 2;
        f = f + ddf_y;
    end
    x = x + 1;
    ddf_x = ddf_x + 2;
    f = f + ddf_x + 1;
if (-img_black(y0+y, x0+x))
    blackpix(n,:) = [y0+y, x0+x];
    n = n + 1;
end
if (-img_black(y0+y, x0-x))
    blackpix(n,:) = [y0+y, x0-x];
    n = n + 1;
end
if (-img_black(y0-y, x0+x))
    blackpix(n,:) = [y0-y, x0+x];
    n = n + 1;
end
if (-img_black(y0-y, x0-x))
    blackpix(n,:) = [y0-y, x0-x];
    n = n + 1;
end
if (-img_black(y0+x, x0+y))
    blackpix(n,:) = [y0+x, x0+y];
    n = n + 1;
end
if (-img_black(y0+x, x0-y))
    blackpix(n,:) = [y0+x, x0-y];
    n = n + 1;
end
if (-img_black(y0-x, x0+y))
    blackpix(n,:) = [y0-x, x0+y];
    n = n + 1;
end
if (-img_black(y0-x, x0-y))
    blackpix(n,:) = [y0-x, x0-y];
    n = n + 1;
end
end
if (n>2)
    dot = mean(blackpix);
elseif (n>1)
    dot = blackpix;
end
if (n>1)
    angle = atan2(dot(1)-y0,dot(2)-x0);
end
end
```c
#define RDYBYTE 0x11
#define FORWARD 0xA
#define PIVOTRIGHT 0x6
#define PIVOTLEFT 0x9
#define UART_TX_BUF_SIZE 50

// Global Variables
OS_STK          TaskStartStk[TASK_STK_SIZE], // Task stacks
MotorTaskStk[TASK_STK_SIZE],
WheelTimerStk[TASK_STK_SIZE];
OS_EVENT         *TransmitSem;
OS_EVENT         *ReadySem;
INT8U            UartTxBuf[UART_TX_BUF_SIZE]; // Transmit buffer
INT16U           UartTxRdPtr; // Transmit read pointer
INT16U           UartTxWpPtr; // Transmit write pointer
INT16U           UartTxCount; // Num of chars left to send
OS_EVENT         *UartTxSem;
INT8U            UartRxBuf[UART_TX_BUF_SIZE]; // Receive buffer
INT16U           UartRxWpPtr; // Receive write pointer
OS_EVENT         *UartRxSem;
INT8U            wheelspeed1 = 0, wheelspeed2 = 0; // Wheel speed targets
//INT8U            wheelTimer = 0;
INT16U           wheelTimer;
OS_EVENT         *WheelTimerSem;

const INT8U     angle[18]  = { 63, 68, 74, 80, 85, 90, 95, 100, 104, 109, 113, 117, 121, 124, 127, 130, 133, 136};

// Prototypes
void TaskStart( void *data );
void PutChar( char c );
void AvrInit( void );
void MotorTask( void *data );
void MotorTimer( void *data );
void sendmsg(INT8U msg);

INT8U CRC8(INT8U* data, int length);

// Main
int main (void)
{
    AvrInit();
    OSInit();

    UartTxSem = OSSemCreate(UART_TX_BUF_SIZE);
    UartRxSem = OSSemCreate(0);
    WheelTimerSem = OSSemCreate(0);
    TransmitSem = OSSemCreate(1);
    ReadySem = OSSemCreate(0);

    OSTaskCreate(TaskStart, (void *)&TaskStartStk[TASK_STK_SIZE - 1], 4);
    OSStart();
    return 0;
}

/* *************** Startup Task *************** */
void TaskStart (void *data)
{
    INT8U err;
    INT8U sequence = 255;
}```
OS_SEM_DATA ReadySemQuery;
INT16U i;

OS_ENTER_CRITICAL();
TCCR0=0x07; //TIMER0 Prescaler = Clock / 1024
TIMSK=_BV(TOIE0); //Enable TIMER0 Interrupt
TCNT0=256-(CPU_CLOCK_HZ/OS_TICKS_PER_SEC/1024); //Reset Counter
OS_EXIT_CRITICAL();

OSTaskCreate(MotorTask, (void *)0, (void *)&MotorTaskStk[TASK_STK_SIZE - 1], 10);
OSTaskCreate(MotorTimer, (void *)0, (void *)&WheelTimerStk[TASK_STK_SIZE - 1], 5);

for (;;)
{
    OSSemPend(UartRxSem, 0, &err); //Wait until a packet is received
    if ((UartRxBuf[1] == ADDRESS) && (UartRxBuf[2] == 0xFF)) //If it's been addressed to us
    {
        if (UartRxBuf[6] == CRC8(UartRxBuf[1], 5)) //If the checksum is OK
        {
            if (UartRxBuf[3] == ACKBYTE)
                OSSemPost(ReadySem);
            else
            {
                OSTimeDly(100);
                sendmsg(ACKBYTE);
                if (UartRxBuf[5] == sequence) //If it's a new packet
                {
                    sequence = UartRxBuf[5];
                        wheelTimer = angle[UartRxBuf[4]];
                    if (UartRxBuf[3] == FORWARD)
                        wheelTimer = forward[UartRxBuf[4]]; //Lookup distance
                    //wheelTimer = (UartRxBuf[3] << 4) | (UartRxBuf[4] & 0x0F); //PORTB = FORWARD;
                    PORTB = UartRxBuf[1]; //Set the direction
                    OSSemQuery(ReadySem, &ReadySemQuery);
                    for (i = 0; i < OS_EVENT_TBL_SIZE; i++)
                    {
                        if (ReadySemQuery.OSEventTbl[i])
                            OSSemPost(ReadySem);
                    }
                    wheelspeed1 = 245; //Turn on both motors
                    wheelspeed2 = 255;
                    OSSemPost(WheelTimerSem); //Signal the motor timer
                }
            }
        }
    }
}

/* *************** Tasks *************** */

void MotorTask( void *data )
{
    for (;;)
    {
        if (OCR1A < wheelspeed1)
            OCR1A += 5;
        if (OCR1A > wheelspeed1)
            OCR1A -= 5;
        if (OCR1B < wheelspeed2)
            OCR1B += 5;
        if (OCR1B > wheelspeed2)
            OCR1B -= 5;
    }
}


```c
OSTimeDlyHMSM(0, 0, 0, 0);

void MotorTimer( void *data )
{
    INT8U err;
    for (;;)
    {
        OSSemPend(wheelTimerSem, 0, &err);
        //OSTimeDlyHMSM(0, 0, wheelTimerS, wheelTimermS);
        OSTMimeDly(wheelTimer);
        wheelspeed1 = 0;
        wheelspeed2 = 0;
        OSTMimeDly(250);
        for (;;)
        {
            sendmsg(RDYBYTE);
            OSSemPend(ReadySem, 300 + (rand() / (RAND_MAX / 200 + 1)), &err);
            if (err == OS_NO_ERR)
                break;
        }
    }
}

void AvrInit( void )
{
    UartT xCount = 0;  // Initialize variables
    UartTxRdPtr = 0;
    UartTxWrPtr = 0;
    UartRxRnPtr = 0;
    DDRA = 0xFF;  //Turn on LED Outputs
    PORTB = FORWARD;  //Set Direction
    TCCR1A = _BV(WGM10) | _BV(COM1A1) | _BV(COM1B1);  //Set up PWM
    TCCR1B = _BV(CS10) | _BV(CS11);  //Set up on motor control outputs
    UBRR1L = F_CPU / 16 / BAUD - 1;  // Set BAUD rate
    UCSR1B = _BV(TXEN1) | _BV(RXEN1) | _BV(RXIE1) | _BV(TXC1);  // Enable USART
    PORTA |= _BV(PA7);  // LED0 On
    UCSR1C = _BV(UCSZ11) | _BV(UCSZ10);  // Set USART transmission params (8 bit, one stop bit)
}

void PutChar( char c )
{
    INT8U err;
    OSSemPend(UartTxSem, 0, &err);  // Wait for Data Register Empty ISR to signal room
    OS_ENTER_CRITICAL();
    UartTxBuf[UartTxWrPtr] = c;
    UartTxWrPtr++;
    UartCount++;
    if (UartTxWrPtr == UART_TX_BUF_SIZE)
        UartTxWrPtr = 0;  // Wrap pointer if necessary
    UCSR1B |= _BV(UDR1E1);
    PORTA |= _BV(PA0);  // Enable the send register empty interrupt
```
OS_EXIT_CRITICAL();
}

void sendmsg(INT8U msg)
{
    INT8U err;
    INT8U packet[7];
    packet[0] = 0xAA;
    packet[1] = 0xFF;
    packet[2] = ADDRESS;
    packet[3] = msg;
    packet[4] = 0x00;
    packet[5] = 0x00;
    packet[6] = CRC8(&packet[1], 5);
    OSSemPend(TransmitSem, 0, &err);
    for(int i = 0; i<10; i++)
        PutChar(0xAA);
    for(int i = 0; i<7; i++)
        PutChar(packet[i]);
    OSSemPost(TransmitSem);
}

INT8U CRC8(INT8U* data, int length)
{
    INT8U CRC = 0xFF;
    for(int i = 0; i < length; i++)
    {
        CRC = CRC ^ data[i];
        for (int j = 0; j < 8; j++)
            CRC = (CRC & 0x80) ? (CRC << 1) ^ 0x1D : (CRC << 1);
    }
    return CRC;
}

/* ***************** Interrupt Service Routines ***************** */

// USART1 Data Register Empty ISR
UCOSISR(USART1_UDRE_vect)
{
    PushRS();
    OSIntEnter();
    if(OSIntNesting == 1)
        OSTCBCur->OSTCBStkPtr = (OS_STK *)SP;
    if(UartTxCount)
    {
        UartTxCount--;
        UCSR1B &=~BV(RXEN1);  
        PORTA &~BV(PA7);    //LED7 Off
        UDR1=UartTxBuf[UartTxRdPtr];
        UartTxRdPtr++;
        OSSemPost(UartTxSem);
        if(UartTxRdPtr==UART_TX_BUF_SIZE)
            UartTxRdPtr=0;
    }
    else
        {  // Nothing to send
            UCSR1B &~BV(UDRIE1);
            PORTA &~BV(PA0);  //LED0 Off
            sei();
            OSIntExit();
            PopRS();
        }
}

//USART1 Data Received ISR
UCOSISR(USART1_RX_vect)
{
    PushRS();
}
OSIntEnter();
if (OSIntNesting == 1)
    OSTCBCur->OSTCBStkPtr = (OS_STK *)SP;

    PORTA |= _BV(PA3);
    UartRxBuf[UartRxWrPtr] = UDR1;                          // Get the char from the buffer
    if(UartRxBuf[UartRxWrPtr]==0xAA)                         // If it's the start of a packet move to the start
    {
        UartRxBuf[0] = 0xAA;
        UartRxWrPtr = 0;
    }

    UartRxWrPtr++;
    if (UartRxWrPtr==UART_TX_BUF_SIZE)                     // Wrap the buffer if we need to
        UartRxWrPtr = 0;
    if((UartRxBuf[0]==0xAA) && UartRxWrPtr > 6)            // If we've received a whole packet's worth
    {
        UartRxBuf[0] = 0;                                  // Prevent processing this packet again
        OSSemPost(UartRxSem);                               // Notify the handler
        PORTA &= ~_BV(PA3);
    }

sei();
OSIntExit();
PopRS();

//USART1 Transmission Complete ISR
UCOSISR(USART1_TX_vect)
{
    PushRS();
    OSIntEnter();
    if (OSIntNesting == 1)
        OSTCBCur->OSTCBStkPtr = (OS_STK *)SP;
    UCSR1B |= _BV(RXEN1);                                  //Transmission ended, enable reception
    PORTA |= _BV(PA7);                                      //LED7 on

    sei();
    OSIntExit();
    PopRS();
}
D. Monash Packet Radio Schematic
E. Sens-r main board schematic
F. Boost Power Supply Schematic