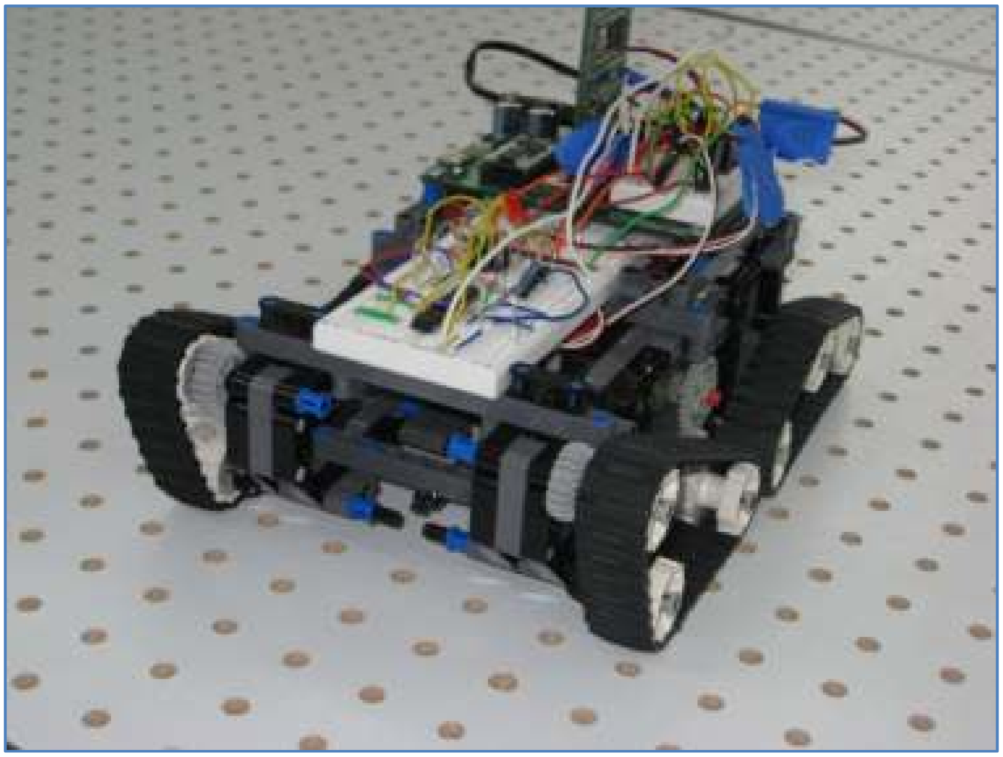
Binford ThermDetector 3000

Coal Mine Fire Detection System



Spring 2009

Team 3: ME5643 Mechatronics Dr. Vikram Kapila
Anthony Bonventre, Bo Deng, Luke Figueroa

Table of Contents

Introduction

Coal mine fires, also known as coal seam fires, are burning deposits of coal. These fires burn long after their surface fires are extinguished. They are either human-induced or naturally occurring. For example, mining operating may ignite the coal. As far as natural occurrences are concerned, this occurs during occasions such as lightning, forest fires, and spontaneous combustion as a result of heat from the sun, or heat from water vapor. In the most extreme cases, coal fires can last for as long as 30 years.

Coal mine fires are dangerous to human life. Toxic gases can be emitted such as Carbon Monoxide (CO), Carbon Dioxide (CO2), Methane (CH4), Sulfur Dioxide (SO2), Nitrous Oxide (N2O), and various Nitrogen Oxides (NOx). These gases affect the quality of life for the surrounding habitants, and their habitations. Emitted aerosols can seep their way into waterways and farmland. Coal mine fires burn the strata underneath the land, turning the layers into ash, creating unsafe living conditions. All in all, the health of people is put at risk under these conditions. Coal mines fires affect the air, the water, and the food supply. For industrially dependent nations that rely heavily on coal, such as China, the risk of unsafe living conditions is especially true. One such coal mining town in Centralia, Pennsylvania had fully evacuate due to a human-induced coal mine fire. The coal mine fire was caused by the burning of a dump, affecting air quality. A young boy, by the name of Todd Domboski, fell into a hole due to the collapse of underlying strata. The dangers that coal mine fires pose to human life should not be underestimated

A solution has been reached through the creation of the Binford ThermDetector 3000: Coal Mine Fire Detection System. By utilizing analog light sensors, a digital IR thermometer, and a Hall-effect sensor, it will detect dangerous levels of heat and changing magnetic fields across a coal mine area. From the subsequent readings, the proper officials can act appropriately, by deciding to extinguish the fire, or in the worst case scenario, evacuate an area in anticipation of collapse, or possible toxic fume emission. The system has been outfitted with a warning system to alert officials, and local citizens when heat and gas levels have exceeded appropriate amounts.

Theory

The Binford ThermDetector 3000: Coal Mine Fire Detection System will have five systems in place: a navigational system employing Parallax QTI light sensors; a thermal system using an infrared thermometer; a warning system requiring the use of a 556 timer IC, LED, and Piezospeaker; a Hall-effect Sensor; and a Bluetooth module to transmit readings to a nearby computer for the generation of a heat map.

The navigational system heavily requires the use of two Parallax QTI sensors. The sensors use a QRD1114 infrared(IR) reflective sensor. When over dark regions, the reflectivity is quite low; while over light regions, the reflectivity would be quite high. Originally, the use of a GPS transmitter and receiver was conceptualized; however GPS modules are rated for resolutions of 2 meters. Because of the relative scale of this project, other means had to found. To resolve this, the uses of light sensors were employed. The area over which the ThermDetector would cover is an area of 80cm x 156cm. This area was taped out on a pegboard, for means that will be explained later. The robot would use a "lawn-mowing" action of sweeping the field. The robot will make quite simple movements, as it will assist in plotting the heat map. The light sensors are programmed to observer for the border of the area, which is shown by the use of black electrical tape. Without looking at the crudity of this exercise, these actions were taken because of the inability to use a GPS module. Because of these setbacks, the team has had to look for alternative means, and the steps toward those means have been properly taken.

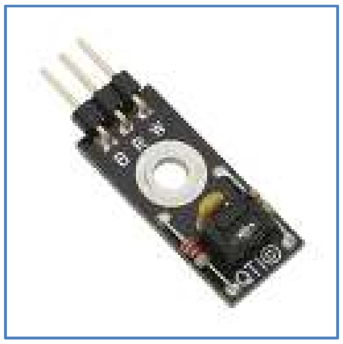


Figure 1: QTI Infrared Light Sensor

The thermal system relies on the use of the MLX90614 Infrared Thermometer Module (90° FOV). The thermometer module is an intelligent non-contact temperature sensor with a 90° field of view, and a serial interface for easy connection to host microcontrollers. When objects are placed within the module's zone of detection, it can obtain accurate surface temperatures with the use of its integrated ASIC and infrared sensitive thermopile detector. As important as movement is to the project, the concept to taking temperature readings with each point on the map is just as important. The thermometer is placed at a height of 1.5cm above the surface, meaning that the system can take surface temperature readings in a 3cm radius, allowing for very effective readings. When taking actual readings to demonstrate, cups of varying temperatures will be placed under the pegboard. A pegboard was chosen to allow for even distribution of heat. Even though the surface temperatures may vary by only a few degrees, coal mine fires burning several hundred or thousand feet below the surface may have adverse effects by raising the temperature by as little as a few degrees. The compounded readings of the traverse area will be sent to a remote computer via Bluetooth to create a heat map.

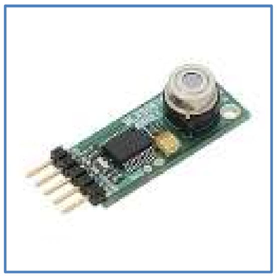


Figure 2: MLX90614 IR Thermometer

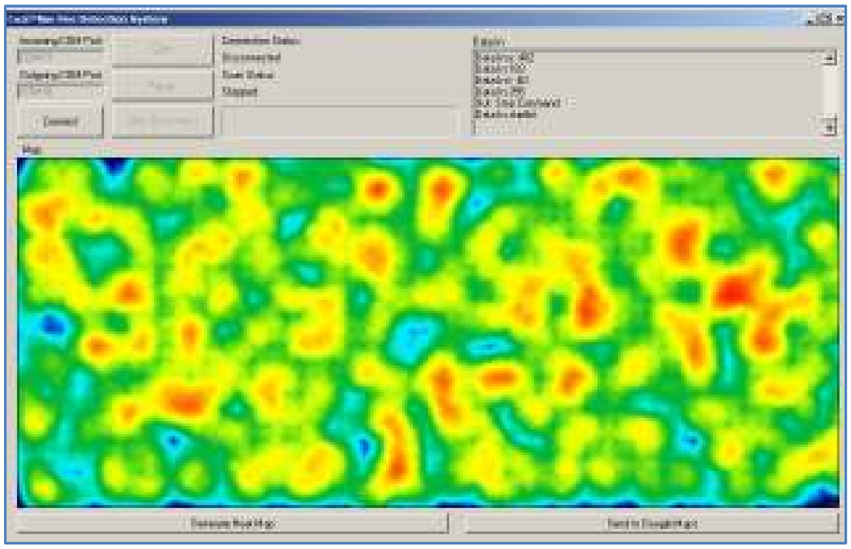


Figure 3: Sample Heat Map

The warning system is composed of 2 main portions, an LED for visual warning and a Piezo speaker for auditory warning. Because both portions are to oscillate at the same frequencies, a 556 dual 16-pin timer is used to offload the software task into a hardware task.

Figure 4: Red LED

Figure 5: Piezo Speaker

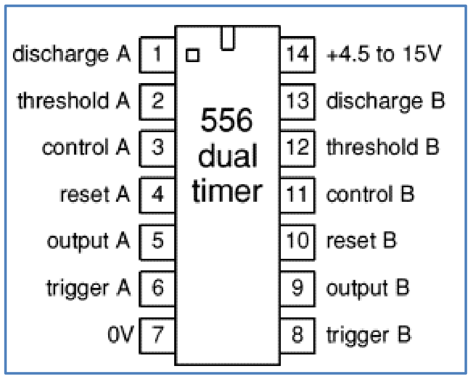


Figure 6: 556 Dual Timer Pin Map

In addition to a search for various changes in surface temperature, a Hall-effect sensor connected through an A/D converter is added. A Hall-effect sensor is a sensor that outputs an analog voltage is proportion to the strength of the applied magnetic field. As an alternative means of sensing the presence of coal mine fires, various geophysical measurements have been created, one such way is by measuring magnetism to determine changes in the magnetic characteristics of the adjacent rock caused by fluctuations in heat. An Allego A1360 Linear Hall-effect sensor was used in connection with an AD0831 A/D Converted. Since the output of the Hall-effect sensor is an analog voltage, the A/D converter is used so that a usable digital voltage signal is obtained.

Figure 7: A1360 Hall-effect Sensor

In order to wirelessly transmit data to a remote computer, a Bluetooth transceiver was added to the ThermDetector. Parallax's own EmbeddedBlue Transceiver AppMod allows for advanced wireless connectivity, by easily plugging into the BS2's pins. This is a step above RF transmission, as it allows full use with all Bluetooth enabled devices. In this manner, data can be transmitted to a remote computer to generate the heat map of the traversed area.

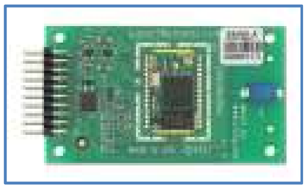


Figure 8: Bluetooth Transceiver

Mechanical Design

A project of this scope would require a robot that can be built easily, as well as modified easily. For this reason, various Lego pieces were used in conjunction with the electrical, and mechanical parts. Two Parallax (Futaba) continuous rotation servomotors, one on each side of the ThermDetector for a 2 motor drive, were mounted by use of Lego parts. The motors were directly connected to the Board of Education in the proper pin locations. The right motor is connected to Pin 14, while the left motor was connected to Pin 15. This allows to individual control, while the position of the motors allows for central axis turning. The servomotor was modified to hold a Lego axle, allowing for use of various sized gears. The servo head was detached, and epoxy was applied to a small Lego gear and attached to the servo head. This allows for full swap-out ability when it comes to choosing proper gear sizes for our needs

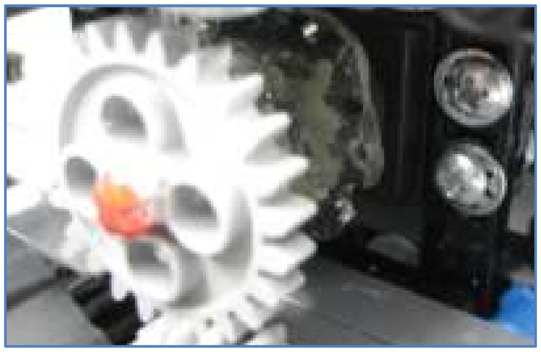


Figure 9: Modified Servomotor

Treads were used for ease of motion. The treads are Lego treads, for nominal ease of use for construction. A gear train was added to transfer motion from the servomotors to the two sides of the ThermDetector. The full design of the ThermDetector is a compact chassis, which includes the Board of Education upon which the BS2 Microcontroller is attached to, as well as the basic bread board to which

Figure 10: Lego Treads

the light sensors and the temperature sensor is connected to. The servos connect to their pins in a separate location. Connected to the Board is a battery casing for four 1.5Volt Batteries. The Bluetooth module is attached to the board, sticking vertically upward. Because of the high level of density of the ThermDetector, the electrical components had to be created with ease of placement in mind.

Electronic Circuits

Warning System

As mentioned earlier, the items used for this circuit are as follows: 1 LED (Red), 1 Piezo speaker, one 556 dual timer IC, one AD5220 digital potentiometer.

The 556 dual timer operates in astable mode. It outputs a square wave so that the voltage is either high or low. A 556 dual timer is used in place of two 555 timers. It serves two purposed: to blick the LEDD and to create a siren with the Piezospeaker. Each side of the 556 timers represents one 555 timer excluding the shared ground and Vdd pins. The LED and speaker were built as they would on a 555 timer. Below are the calculations for the LEDs, R1, R2, and C from the desired t_{high}, t_{low}, and desired frequency values. The capacitance value was chosen to be 10µF so as to automatically bring down the LED frequency and make the blinking visible. The value for t_{high} was chosen to be about 100ms so that it would be visible and t_{low} to be 200ms.

$$t_{high} = 0.693R_2C \Rightarrow R_2 = \frac{t_{high}}{0.693C} = \frac{100ms}{0.693 * 10{,}000nF} = 14430$$

$$R_2 \text{ is chosen to be } 15k\Omega$$

$$t_{low} = 0.693(R_1 + R_2)C \Rightarrow R_1 = \frac{t_{low}}{0.693C} - R_2 = \frac{200ms}{0.693 * 10{,}000nF} - 15{,}000 = 13860$$

$$R_1 \text{ is chosen to be } 10k\Omega$$

$$f = \frac{1}{t_{high} + t_{low}} = \frac{1}{0.693C(R_1 + 2R_2)} = \frac{1}{(0.693)(10{,}000nF)(10k\Omega + 2 * 15k\Omega} = 3.6Hz$$

The frequency is roughly 3.6Hz, which is visible.

Since the speaker is only audible at higher frequencies (3200 Hz or above), the R1, R2, and C values were chosen to create higher frequencies. Then Output B (Pin 9) is connected to Discharge A (Pin 1) with a 10kΩ so that the speaker frequency is controlled by another frequency. Thus, the speaker is audible since it is at a higher frequency, but that speaker oscillates at a lower frequency. Physically speaking, the speaker would be loud but would go on and off. Here, the speaker would oscillate at the same rate as the LED because it is connected to the same output as the LED.

The R1, R2, and C were chosen based on the graph on the right. C was chosen to be 0.01µF so that it would oscillate quickly. To be in the 1kHz < f <10kHz ranger, R1 + 2R2 was chose to be slightly larger than 10kΩ. R1 and R2 were chosen to be 10kΩ and 5kΩ respectively.

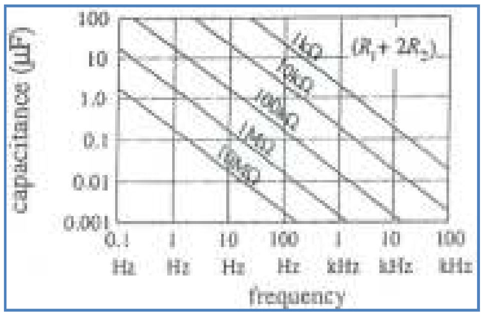


Figure 11: Capacitance vs. Frequency

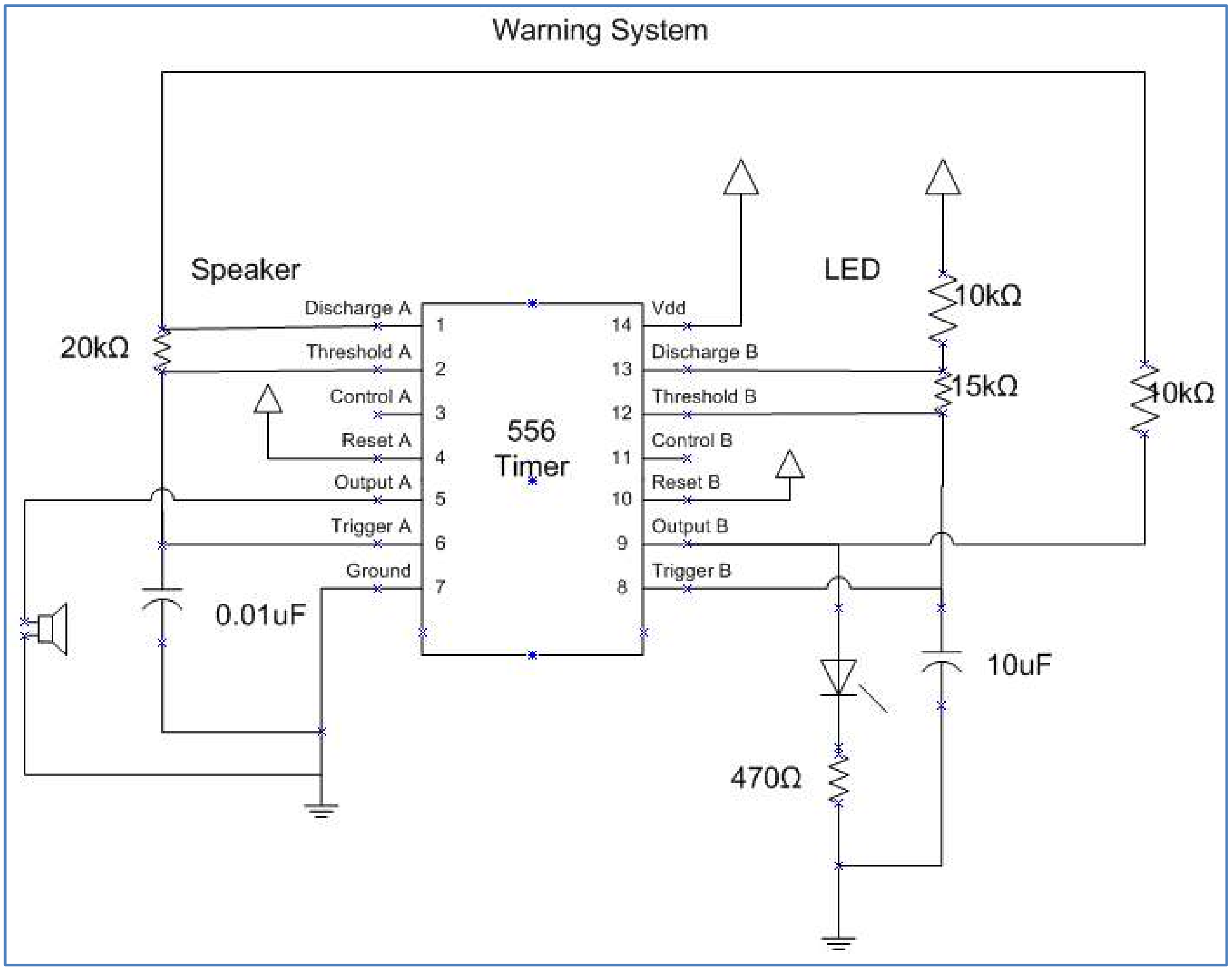


Figure 12: Warning System Schematic

Hall Effect Sensor

The Hall-effect Sensor used was an Allegro A1360 Linear Hall-effect Sensor. In order to find the working range, a gaussmeter was built according the following diagram:

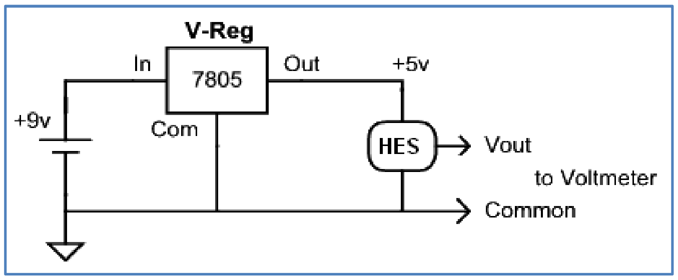


Figure 13: Gaussmeter Schematic

With this, various voltages can be obtained from the Hall-Effect Sensor. Because the output of the Hall-effect sensor is an analog voltage, this can be taken advantage of by connecting it to a voltmeter. Due to the small sensitivity of the device, a range of 0.7 to 1.4 mV/G, the voltage changes went from 1.98V when little or no magnetic field present, to a maximum value of 2.51V, and minimum value of 1.45V. The proper poles of an applied magnetic field can be determined with the following equation:

$$B = \frac{1000(V_0 - V_1)}{k}$$

Where:

V_0= Voltage under the presence of little or no magnetic field

V_1=Voltage measured from voltmeter under presence of a magnet.

B= Magnetic Flux Density, in Gauss

k= sensitivity of the Hall-effect sensor, taken to be 1.05mV/G

Taking two different readings we obtain:

$$B_a = \frac{1000(V_0 - V_{1a})}{k} = \frac{1000(1.98 - 2.51)}{1.05} = -504.762 Gauss, South\ Pole\ (Negative)$$

$$B_b = \frac{1000(V_0 - V_{1b})}{k} = \frac{1000(1.98 - 1.45)}{1.05} = 504.762 Gauss, North\ Pole\ (Positive)$$

To obtain these values, t he magnet was placed as close as possible to the Hall-effect sensor in order to get the largest magnitudes for the values. It is worthwhile to note that the values are the same in magnitude, only differing in sign.

Once those values have been obtained and noted down, it is time to connect it to an A/D converter to take the previously analog voltages to obtain digital values from them. Pins 1, 6, and 7 were connected to 3 Pins of the BS2. Of the ADC0831, when Pin 1 is driven low A2D conversion is ready to happen, Pin 6 is where the 8bit A2D output will come from, and Pin 7 is where the

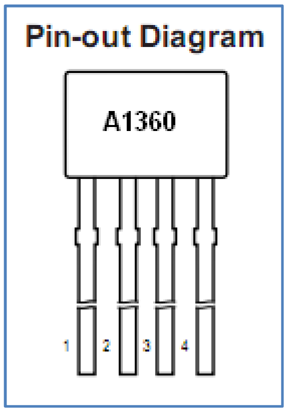


Figure 14: Hall-effect Sensor Pin Map

Clock signal from BS2 is received. Pins 5 and 8 are driven high to 5Volts. Pins 3 and 4 are driven low to Vss. Pin 2 of the ADC0831 is where the analog input will be received. This analog input is the one that will be digitized. Pin 2 of the ADC0831 is connected to Pin 2 of the Hall-effect sensor. Pin 1 of the Hall-effect sensor will be driven high to 5 volts, while Pin 4 of the Hall-effect sensor will be driven low to Vss. Pin 3 has no connection as Pin 3 is the terminal used for external filter capacity for bandwidth setting; this has no relevance for the purposes that we are using the Hall-effect sensor for the project. Once all the connections have been made, the Hall-effect sensor circuit was placed on a breadboard and mounted so the proper angles required for full sensing are used.

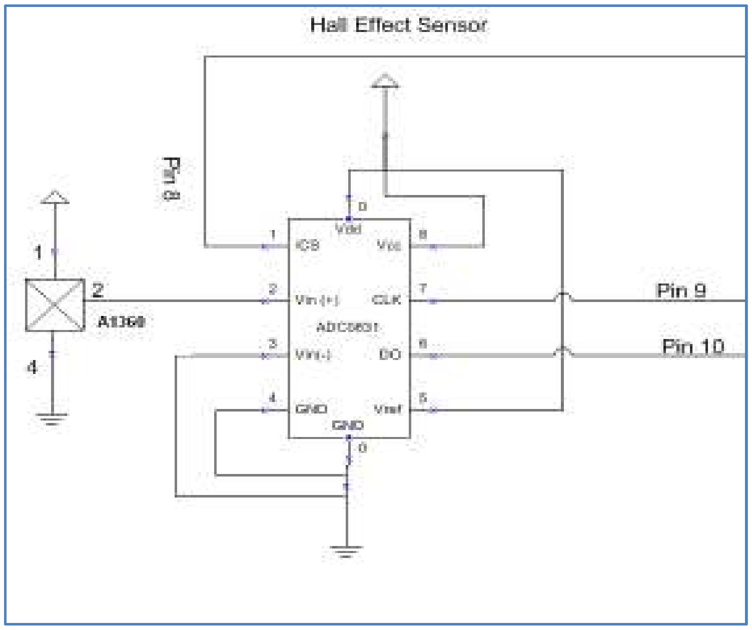


Figure 15: Hall Effect Sensor Schematic

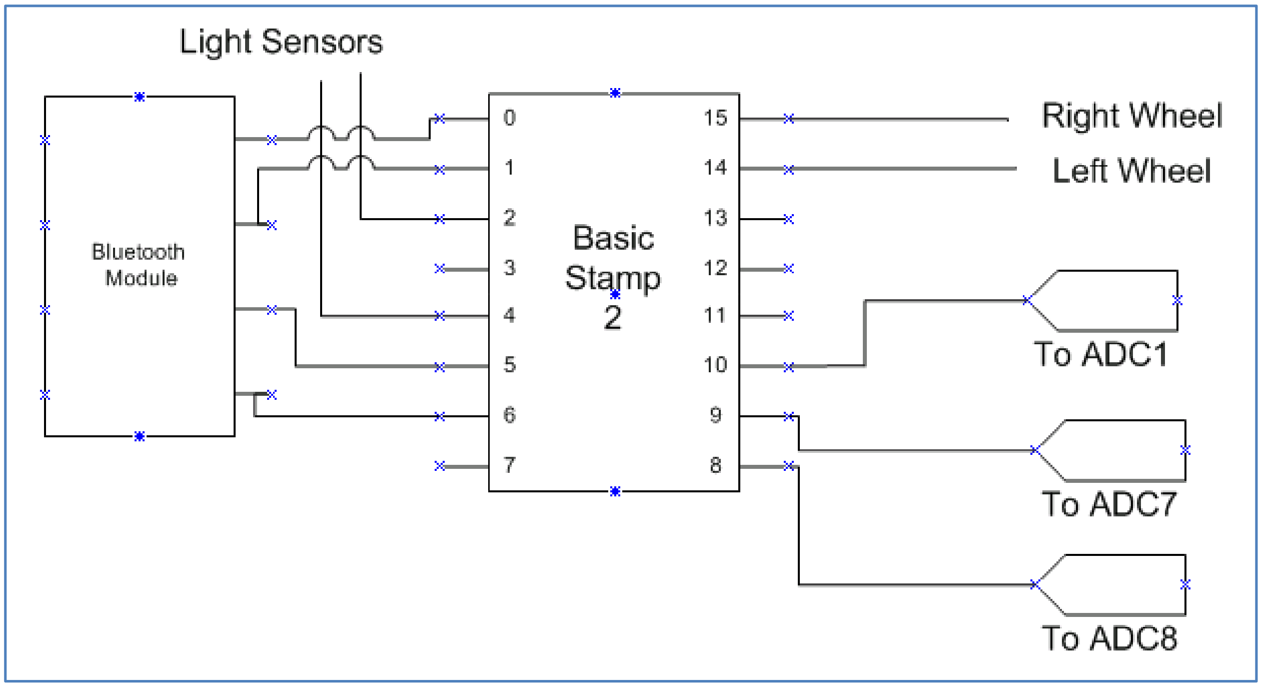


Figure 16: Basic Stamp Schematic

Bill of Materials

Prototype Cost

Bill of Materials: Prototype			
Item	Quantity	Cost	Total
Basic Stamp 2 Module	1	$49.00	$49.00
Board of Education Development Board (USB)	1	$69.99	$69.99
Parallax (Futaba) continuous rotation servomotors	2	$12.99	$25.98
QRD1114 infrared(IR) reflective sensor	2	$5.99	$11.98
MLX90614 Infrared Thermometer Module (90° FOV)	1	$39.99	$39.99
EmbeddedBlue Transceiver AppMod	1	$69.99	$69.99
556 Timer	1	$1.99	$1.99
Allegro A1360 Linear Hall-effect sensor	1	$2.57	$2.57
ADC0831 A2D converter	1	$5.99	$5.99
Lego Parts	1	$20.00	$20.00
Various electrical components (LEDs, Capacitors, Resistors, wires, etc)	1	$30.00	$30.00
Battery holder	1	$1.99	$1.99
Pegboard Map	1	$20.00	$20.00
Total Cost			**$349.47**

Table 1: Cost of One Prototype Unit

Mass Production Cost

Bill of Materials: Cost for Mass Production (100 units)			
Item	Quantity	Cost	Total
Basic Stamp 2 Module	1	$39.20	$39.20
Board of Education Development Board (USB)	1	$55.99	$55.99
Parallax (Futaba) continuous rotation servomotors	2	$11.69	$23.38
QRD1114 infrared(IR) reflective sensor	2	$5.99	$11.98
MLX90614 Infrared Thermometer Module (90° FOV)	1	$39.99	$39.99
EmbeddedBlue Transceiver AppMod	1	$69.99	$69.99
556 Timer	1	$1.99	$1.99
Allegro A1360 Linear Hall-effect sensor	1	$1.60	$1.60
ADC0831 A2D converter	1	$4.79	$4.79
Lego Parts	1	$20.00	$20.00
Various electrical components (LEDs, Capacitors, Resistors, wires, etc)	1	$30.00	$30.00
Battery holder	1	$1.99	$1.99
Total Cost			**$300.90**

Table 2: Cost of One Mass Produced Unit

Advantages & Disadvantages

Advantages of Design

A main advantage of this system is t hat it is a cheap alternative to thermal graphic imaging and remote sensing. In remote sensing, a satellite will detect the gas or heat emitted from coal mine fires. However it may not be reliable during the daytime as the sun also produces heat. This robot will be able to map out the territory more reliably during the daytime. Particularly, the robot serves locally as a warning system. This is something that satellite systems will not be able to do without interaction with a communication system.

The mechanical design of the ThermDetector is something to take notice of when considered products of this nature. Because of the treads used in its design, with higher powered actuators, the ThermDetector will be able to overcome many obstacles in its way. Slight inclines will be no problem, as well as on point turning, as the system is able to do so when programmed to. All electrical components are all in one position. In an interconnecting weave, the multiple subsystems are are placed on the same breadboard to reduce cost and wasted space. Despite this, the system still does its intended duty.

A notable advantage is the Hall-effect sensor to sense change in magnetic properties of rock and ground. Armed with this knowledge, this is another method of searching for possible heat sources. Because many of the same methods have been employed when it comes to searching for coal mine fires, allowing the use of an unorthodox method may be the one that effectively saves lives

Problems Encountered

The Basic Stamp microcontroller and its connecting parts are not perfect. Over the many iterations and testing phases of the system, problems were encountered. These problems were all either routed, or simply taken at face value.

One such problem, as mentioned earlier was the resolution of the GPS transceiver. Because of the large resolution (approximately 2m), it could not be used properly in the scale of our area (approximately 80cm x 156 cm). The GPS receiver would have been a larger problem than a solution. The method that was taken instead was to use 3 QTI light sensors, which are quite accurate, in conjunction with an absolute position map, to create a logic system simulating longitude and latitude. The design of the map was as follows: the first line was a horizontal line of increasing gradient from near black to just about gray, the line that followed was a solid line of a certain grayscale value, the third line was the same as the first, the line that followed was a solid line of certain grayscale value higher than the 2nd line. With this logic, and continuing for about 60+ lines, light sensor readings would be able to give us values that would allow us to correctly map out the system's position in the X and Y direction. A bonus from using an absolute position system would allow us to plot the temperature map with greater ease, as each point in 2D space would hold 3 values: X position, Y position, and temperature reading. However, due to programming woes, this could not be done. The obtained map from Kinko's was printed with large imperfections, such as various lines not being the grayscale value not asked for, streaks appearing on the lines from natural plotting techniques. Because of these problems, the absolution position map was abandoned in favor for another possibility.

The 2nd possibility was to place a tight zig-zag pattern on the pegboard, which the robot would theoretically follow for the area of the map. At this point, the light sensor, still being used in an analog manner was creating many issues for us, such as creating conflicting cases, and not fitting in the proper spot. The 2nd possibility narrowed down the use of light sensors from 3 to 2. This possibility failed quickly when the turning logic in the program simply was not doing what it was told. Rather than waste time by spending it in the loop of continually testing and reprogramming, the final possibility was approached.

The course, as it runs now is a simple 80cm x 156cm rectangular area bordered by black electrical tape, over a pegboard. As mentioned early, the purpose of the pegboard will allow for heat to disperse properly. The black electrical tape acts as the border, allowing the system to know the boundaries of the land given to it to plot. Once the light sensors see the border, the ThermDetector will turn around to continue its scanning over the rest of the plot of land. This final iteration works very well, as low amounts of error has been seen from the numerous testing involved.

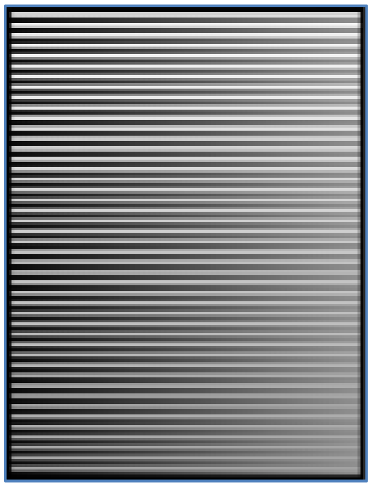


Figure 17: Absolute Map to be Used for 1st Iteration of Navigation System

A gas sensor was originally planned for use. This idea was thoroughly followed through. A small shaft of 50mm, and 6.35mm in diameter was obtained in order for the gas sensor to work. The shaft was drilled through with 2mm diameter holes, placed 3mm apart center to center. This was done quite beautifully, and all hopes were high. In order for the gas sensor to work an IR LED and an IR Detector were placed on opposite ends of the shaft. With both components fitting exactly in the shaft, the ends were sealed with liquid electrical tape to ensure that no interference from the outside would make it inside the shaft.

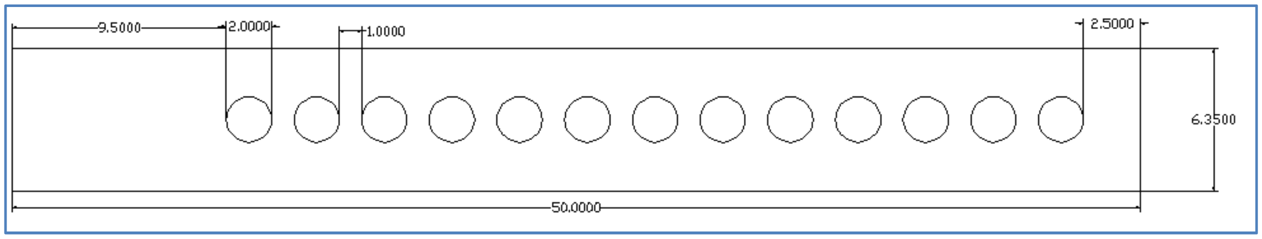


Figure 18: Gas Sensor Originally Planned For Use

When it came to the testing portion of the sensor, it failed. No conclusive results were found when tested using Nitrous Oxide, a clear gas that is non-toxic. A non-toxic gas was chosen because when it came time to demonstrate the project, effectively risking the lives of fellow classmates is not what this team wants to be known for. One of the foreseeable possible issues with a sensor such as this is that if Gas A (toxic) is tested to have a certain value, how can we judiciously know that when we view a value that point s to Gas A (toxic), that it's not really Gas B (safe). Because of this problem, a procedure such as alerting authorities would not help as it would be a severe waste of time and resources. A sensor could still be produced to detect for a range of gases, but gases come in all shapes, colors, and properties. A sensor that uses light wouldn't work properly as many things can look like other things. This was found when testing with Nitrous Oxide. The ambient air was giving a certain value, while when the gas was applied; no change was seen in the result. Although that was only one test, it really only points to 2 things, either the test should be re-done using different gases, or the gas sensor was built incorrectly. The test, if done using different gases, such as those producing visible impurities in the air, would be counter-productive, as it can still be seen by human eye, the point of having such a feature on the ThermDetector would be beaten by human eyes. Adversely, the human involvement in creating the gas sensor has effect as well. Since there were no other sample circuits to follow when creating this sensor, varying techniques were used. None producing a viable outcome. Because of the time constraints this sensor was abandoned to work on other portions of the project.

Conclusion

The Binford ThermDetector 3000 is a very versatile robot. Not only will it be able to traverse a plot of land, but it can also wirelessly send data to a remote station to allow for quick upload, and instant heat map creation. For the purpose of this project, taking that heat map and scaling it properly over a fixed area in Pennsylvania (chosen for its high volume of coal mining towns), will allow us to fully experience the scope of this machine.

Being allowed to look for coal mine fires in a different way is quite important too, for when all methods have failed it is important to think outside the box. What you have just witnessed here is simply a prototype, not ready for the real world, but with added time, and concentration on future iterations, the Binford ThermDetector 3000: Coal Mine Fire Detection System might just save a few lives.

Due to the large amount of setbacks that came while building this machine, the team producing it can readily say that it was not an easy project, but the final joy and bringing it all together was worth it. From the culmination of the project, much can be taken away from this. One such lesson to be learned is that all things that are designed are subject to fail. Through the course of completing this project the design process has had to be started up over and over again in order to try and combat the issues facing us. Not all things that you can design can be made. Before going forward with the testing phase, all things in theory should work, but theory is always far from the applications in real world.

Appendices

Code

Simulation Code

This code is used for showing that the ThermDetector is able to take temperature readings while assigning an X and a Y value to a temperature reading.

```
' {$STAMP BS2}
' {$PBASIC 2.5}

'SIMULATION READINGS
'Had to be used since bluetooth could not be operated
'while the robot was moving, this simulates x and y values
'while taking the surface temperature and sending it to the
'computer to be graphed


' Set variables
bData       VAR    Byte
counter     VAR    Byte
countery    VAR    Byte
temp        VAR    Word
temperature  VAR    Word
tempL       VAR    temperature.LOWBYTE
tempH       VAR    temperature.HIGHBYTE
pec         VAR    Byte
w           VAR    Byte
x           VAR    Byte
y           VAR    Byte
z           VAR    Byte

'Set EEPROM for IR Thermometer
PEC0   DATA $00,$07,$0E,$09,$1C,$1B,$12,$15,$38,$3F,$36,$31,$24,$23,$2A,$2D
PEC1   DATA $70,$77,$7E,$79,$6C,$6B,$62,$65,$48,$4F,$46,$41,$54,$53,$5A,$5D
PEC2   DATA $E0,$E7,$EE,$E9,$FC,$FB,$F2,$F5,$D8,$DF,$D6,$D1,$C4,$C3,$CA,$CD
PEC3   DATA $90,$97,$9E,$99,$8C,$8B,$82,$85,$A8,$AF,$A6,$A1,$B4,$B3,$BA,$BD
PEC4   DATA $C7,$C0,$C9,$CE,$DB,$DC,$D5,$D2,$FF,$F8,$F1,$F6,$E3,$E4,$ED,$EA
PEC5   DATA $B7,$B0,$B9,$BE,$AB,$AC,$A5,$A2,$8F,$88,$81,$86,$93,$94,$9D,$9A
PEC6   DATA $27,$20,$29,$2E,$3B,$3C,$35,$32,$1F,$18,$11,$16,$03,$04,$0D,$0A
PEC7   DATA $57,$50,$59,$5E,$4B,$4C,$45,$42,$6F,$68,$61,$66,$73,$74,$7D,$7A
PEC8   DATA $89,$8E,$87,$80,$95,$92,$9B,$9C,$B1,$B6,$BF,$B8,$AD,$AA,$A3,$A4
PEC9   DATA $F9,$FE,$F7,$F0,$E5,$E2,$EB,$EC,$C1,$C6,$CF,$C8,$DD,$DA,$D3,$D4
PEC10  DATA $69,$6E,$67,$60,$75,$72,$7B,$7C,$51,$56,$5F,$58,$4D,$4A,$43,$44
PEC11  DATA $19,$1E,$17,$10,$05,$02,$0B,$0C,$21,$26,$2F,$28,$3D,$3A,$33,$34
PEC12  DATA $4E,$49,$40,$47,$52,$55,$5C,$5B,$76,$71,$78,$7F,$6A,$6D,$64,$63
PEC13  DATA $3E,$39,$30,$37,$22,$25,$2C,$2B,$06,$01,$08,$0F,$1A,$1D,$14,$13
```

```
PEC14  DATA $AE,$A9,$A0,$A7,$B2,$B5,$BC,$BB,$96,$91,$98,$9F,$8A,$8D,$84,$83
PEC15  DATA $DE,$D9,$D0,$D7,$C2,$C5,$CC,$CB,$E6,$E1,$E8,$EF,$FA,$FD,$F4,$F3

'Set Cosntants
therm       CON    7
baud        CON    84
xslave      CON    $35

'Initialize
counter = 1
countery = 1

'Wait for bluetooth module to initialize
PAUSE 1000

'Wait for connecter to be made
WaitForConnection:
  IF IN5 = 0 THEN WaitForConnection


DEBUG "Connection established",CR

Main:
PAUSE 300
  GOSUB ReadTemp     'Read the temperature
  DEBUG HOME, ? temp, CLREOL
 IF counter > 156 THEN  'Simulate x and y values using counters
 counter = 1
 countery = countery + 3
 ELSEIF counter < 156 THEN
 counter = counter + 3
 ELSEIF countery > 80 THEN
 END
 ENDIF
SERIN 0,84, [STR bData\1]     'Check for data being sent it
SEROUT 1, 84, [255, counter, countery, temp]  'Send data starting with a start signal of 255
GOTO Main


'---------READING SURFACE TEMPERATURE-------------
ReadTemp:
 SEROUT therm,baud,[0,"!TEMR",xslave,$07]
 SERIN therm,baud,1000,PECfail,[tempL,tempH,pec]
  z=0
  y = xslave<<1+0
  GOSUB calculateCRC
  y = $07
  GOSUB calculateCRC
  y = xslave<<1+1
  GOSUB calculateCRC
  y = tempL
  GOSUB calculateCRC
  y = tempH
  GOSUB calculateCRC
  IF z<>pec THEN PECfail
  temp = (temperature/100*2)-273
RETURN

calculateCRC:
  w=z^y
  READ w,z
RETURN

PECfail:
  DEBUG CRSRXY, 23, 3,CLREOL,CR
  DEBUG CRSRXY, 23, 4,CLREOL,CR
  DEBUG CRSRXY, 23, 5,"Fail",CLREOL,CR
  PAUSE 500
GOTO ReadTemp
'-----------------------------------------
```

Navigation Code

This code is used for the main motion of the robot.

```
' {$STAMP BS2}
' {$PBASIC 2.5}

'Set variables
timeR       VAR    Word
timeL       VAR    Word
counter     VAR    Word
turncounter  VAR    Byte
turndir     VAR    Bit
ycount      VAR    Byte
xcount      VAR    Byte
bdata       VAR    Byte
temp        VAR    Word
temperature  VAR    Word
tempL       VAR    temperature.LOWBYTE
tempH       VAR    temperature.HIGHBYTE
pec         VAR    Byte
w           VAR    Byte
x           VAR    Byte
y           VAR    Byte
z           VAR    Byte
adcBits     VAR    Byte
```

```
v          VAR    Byte
r          VAR    Byte
v2          VAR    Byte
v3          VAR    Byte

'Set Constants
Llight     CON 4
Rlight     CON 2
Clight     CON 3
rwheel     CON 14
lwheel     CON 15
therm      CON 7
baud       CON 84
xslave     CON $35
CS        PIN 8
CLK       PIN 9
DataOutput PIN 10

'Set EEPROM for IR Thermometer
PEC0   DATA $00,$07,$0E,$09,$1C,$1B,$12,$15,$38,$3F,$36,$31,$24,$23,$2A,$2D
PEC1   DATA $70,$77,$7E,$79,$6C,$6B,$62,$65,$48,$4F,$46,$41,$54,$53,$5A,$5D
PEC2   DATA $E0,$E7,$EE,$E9,$FC,$FB,$F2,$F5,$D8,$DF,$D6,$D1,$C4,$C3,$CA,$CD
PEC3   DATA $90,$97,$9E,$99,$8C,$8B,$82,$85,$A8,$AF,$A6,$A1,$B4,$B3,$BA,$BD
PEC4   DATA $C7,$C0,$C9,$CE,$DB,$DC,$D5,$D2,$FF,$F8,$F1,$F6,$E3,$E4,$ED,$EA
PEC5   DATA $B7,$B0,$B9,$BE,$AB,$AC,$A5,$A2,$8F,$88,$81,$86,$93,$94,$9D,$9A
PEC6   DATA $27,$20,$29,$2E,$3B,$3C,$35,$32,$1F,$18,$11,$16,$03,$04,$0D,$0A
PEC7   DATA $57,$50,$59,$5E,$4B,$4C,$45,$42,$6F,$68,$61,$66,$73,$74,$7D,$7A
PEC8   DATA $89,$8E,$87,$80,$95,$92,$9B,$9C,$B1,$B6,$BF,$B8,$AD,$AA,$A3,$A4
PEC9   DATA $F9,$FE,$F7,$F0,$E5,$E2,$EB,$EC,$C1,$C6,$CF,$C8,$DD,$DA,$D3,$D4
PEC10  DATA $69,$6E,$67,$60,$75,$72,$7B,$7C,$51,$56,$5F,$58,$4D,$4A,$43,$44
PEC11  DATA $19,$1E,$17,$10,$05,$02,$0B,$0C,$21,$26,$2F,$28,$3D,$3A,$33,$34
PEC12  DATA $4E,$49,$40,$47,$52,$55,$5C,$5B,$76,$71,$78,$7F,$6A,$6D,$64,$63
PEC13  DATA $3E,$39,$30,$37,$22,$25,$2C,$2B,$06,$01,$08,$0F,$1A,$1D,$14,$13
PEC14  DATA $AE,$A9,$A0,$A7,$B2,$B5,$BC,$BB,$96,$91,$98,$9F,$8A,$8D,$84,$83
PEC15  DATA $DE,$D9,$D0,$D7,$C2,$C5,$CC,$CB,$E6,$E1,$E8,$EF,$FA,$FD,$F4,$F3

'Initializations
turndir = 0
xcount = 0
ycount = 1

'Wait for bluetooth to initialize
PAUSE 1000

'Wait for connection to be made
WaitForConnection:
  IF IN5 = 0 THEN WaitForConnection
```

```
DEBUG "CONNECTED"

'Views incoming bluetooth data and waits for
'the start command "S" To be received
waitforstart:
SERIN 0,84, [STR bData\1]
DEBUG CLS, STR bData
IF bdata = "S" THEN
GOTO main
ENDIF
GOTO waitforstart

'Main operation for movement and measurement
main:
  GOSUB readingval  'Get Light Sensor Values
  GOSUB ReadTemp    'Get Temperature
  GOSUB Readhall    'Get Magnet Reading
  DEBUG HOME, ? temp, CLREOL
  'IF IN5 = 0 THEN GOSUB connect
  IF timel<1000 AND timer<1000 THEN   'Check for when both sensors detect white
    GOSUB forward                     'Go forward
    xcount = xcount + 1               'Add 1 to x for location
    DEBUG HOME, ? xcount, CR, ?ycount

 '-----Conditionals for dealing with Navigation-------
  ELSEIF timel>1000 AND timer>1000 THEN
    IF turndir = 0 THEN
      GOSUB turnleft
      turndir = 1
      xcount = 0       'After turn reset x to 0
      ycount = ycount + 1  'Increase y by 1
    ELSEIF turndir = 1 THEN
      GOSUB turnright
      turndir = 0
      xcount = 0
      ycount = ycount + 1
    ENDIF
  ELSEIF timeL < 1000 AND timeR > 1000 THEN
      FOR counter = 1 TO 3
      PULSOUT lwheel, 1000
      PULSOUT rwheel, 800
      PAUSE 20
      NEXT
  ELSEIF timeL > 1000 AND timeR < 1000 THEN
      FOR counter = 1 TO 3
      PULSOUT rwheel, 500
      PULSOUT lwheel, 700
      PAUSE 20
    NEXT
  ENDIF
  '--------------------------------------
GOTO main

connect:
SEROUT 1,84,["con 00:1e:4c:ff:60:82",CR]
SERIN 0,84,[WAIT("ACK",CR)]
IF IN5 = 0 THEN GOTO connect
DEBUG "CONNEC TRY", CR
RETURN

bluetooth:
SERIN 0,84, [STR bData\1]
DEBUG CLS, STR bData
RETURN

'Use RCTIME to read light sensor values
readingval:
HIGH Llight
RCTIME Llight, 1, timeL
HIGH Rlight
RCTIME Rlight, 1, timeR
'DEBUG CLS, ? timeL, CR, ? timeR, CR
RETURN

Readhall:
  LOW CLK
  LOW CS
  PULSOUT CLK, 210
  SHIFTIN DataOutput,CLK,MSBPOST,[adcBits\8]
  HIGH CS
  v = 5 * adcBits / 255
  r = 5 * adcBits // 255
  v2 = 100 * r / 255
  v3 = 100 * r //255
  v3 = 10 * v3 / 255
  IF (v3 >= 5) THEN v2 = v2 + 1
  IF (v2 >=100) THEN
    v = v + 1
    v2 = 0
  ENDIF
  IF v>=2 OR NOT v2=98 THEN
    HIGH 12 ' triggers 556 timer circuit
  ELSE
    LOW 12
  ENDIF
RETURN
```

```
forward:
FOR counter= 1 TO 2
PULSOUT lwheel, 1000
PULSOUT rwheel, 500
PAUSE 20
NEXT
RETURN

turn:
FOR counter = 1 TO 3
PULSOUT rwheel, 500
PULSOUT lwheel, 500
PAUSE 20
NEXT

RETURN

turnreverse:
FOR counter = 1 TO 3
PULSOUT rwheel, 1000
PULSOUT lwheel, 1000
PAUSE 20
NEXT

RETURN

'********TURN LEFT CODE*********

turnleft:

  GOSUB forward
  GOSUB forward


    FOR turncounter = 1 TO 64
      GOSUB turn
    NEXT

  DO
      GOSUB readingval
    IF timeL < 1000 AND timeR < 1000 THEN
      GOSUB forward
    ELSEIF timeL < 1000 AND timeR > 1000 THEN
      FOR counter = 1 TO 2
      PULSOUT lwheel, 1000
      PULSOUT rwheel, 800
      PAUSE 20
      NEXT
    ELSEIF timeL > 1000 AND timeR < 1000 THEN
      FOR counter = 1 TO 2
      PULSOUT rwheel, 500
      PULSOUT lwheel, 700
      PAUSE 20
      NEXT
    ENDIF
  LOOP UNTIL timer > 1000 AND timel > 1000

  DO
    GOSUB readingval
    IF timeL > 1000 AND timeR > 1000 THEN
      GOSUB forward
    ELSEIF timeL > 1000 AND timeR < 1000 THEN
      FOR counter = 1 TO 2
      PULSOUT lwheel, 1000
      PULSOUT rwheel, 800
      PAUSE 20
      NEXT
    ELSEIF timeL < 1000 AND timeR > 1000 THEN
      FOR counter = 1 TO 2
      PULSOUT rwheel, 500
      PULSOUT lwheel, 700
      PAUSE 20
      NEXT
    ENDIF
  LOOP UNTIL timer < 1000 AND timel < 1000


RETURN

'********TURN RIGHT CODE*********

turnright:
  GOSUB forward
  GOSUB forward

    FOR turncounter = 1 TO 64
      GOSUB turnreverse
    NEXT

  DO
      GOSUB readingval
    IF timeL < 1000 AND timeR < 1000 THEN
      GOSUB forward
    ELSEIF timeL < 1000 AND timeR > 1000 THEN
```

```
      FOR counter = 1 TO 2
      PULSOUT lwheel, 1000
      PULSOUT rwheel, 800
      PAUSE 20
      NEXT
    ELSEIF timeL > 1000 AND timeR < 1000 THEN
      FOR counter = 1 TO 2
      PULSOUT rwheel, 500
      PULSOUT lwheel, 700
      PAUSE 20
      NEXT
    ENDIF
  LOOP UNTIL timer > 1000 AND timel > 1000

  DO
    GOSUB readingval
    IF timeL > 1000 AND timeR > 1000 THEN
      GOSUB forward
    ELSEIF timeL > 1000 AND timeR < 1000 THEN
      FOR counter = 1 TO 2
      PULSOUT lwheel, 1000
      PULSOUT rwheel, 800
      PAUSE 20
      NEXT
    ELSEIF timeL < 1000 AND timeR > 1000 THEN
      FOR counter = 1 TO 2
      PULSOUT rwheel, 500
      PULSOUT lwheel, 700
      PAUSE 20
      NEXT
    ENDIF
  LOOP UNTIL timer < 1000 AND timel < 1000


RETURN

'---Get temperature reading for IR Thermometer----
ReadTemp:
 SEROUT therm,baud,[0,"!TEMR",xslave,$07]
 SERIN therm,baud,1000,PECfail,[tempL,tempH,pec]
  z=0
  y = xslave<<1+0
  GOSUB calculateCRC
  y = $07
  GOSUB calculateCRC
  y = xslave<<1+1
  GOSUB calculateCRC
  y = tempL
  GOSUB calculateCRC
  y = tempH
  GOSUB calculateCRC
  temp = (temperature/100*2)-273
RETURN

calculateCRC:
  w=z^y
  READ w,z
RETURN

PECfail:
  DEBUG CRSRXY, 23, 3,CLREOL,CR
  DEBUG CRSRXY, 23, 4,CLREOL,CR
  DEBUG CRSRXY, 23, 5,"Fail",CLREOL,CR
  PAUSE 500
GOTO ReadTemp
'-------------------------------------------------
```

Heat Map Software Code

This piece of code takes data readings from the ThermDetector and uses C code to create the desired heatmap.

```
//   Coal Mine Fire Detection System
//     Binford ThermoDetect 3000
//            Mechatronics
//             Spring 2009
//         Anthony Bonventre
//           Luke Figueroa
//              Bo Deng
// --------------------------------
//   Bluetooth and Mapping Software
// Developed using assistance from:
//   Dylan Vester from C# Playground
//                and
//    Badea Florin from CodeProject
// --------------------------------

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;

namespace HeatMap
{

    public partial class Form1 : Form
    {
        //Set Serial Bluetooth Ports
        //SerialPort serialIn;
        SerialPort serialOut;
        //Set Thread for Receiving Data
        Thread receiveThread;
        int threads = 0;
        //Set private voids for functions
        private delegate void updateText(string s);
        private delegate void closeDel();
        private delegate void disconnectDel();
        private List<HeatPoint> HeatPoints = new List<HeatPoint>();
        //Initialize close and disconnect values to false
        bool closeRequested = false;
        bool disconnectRequested = false;
        int xdata;
        int ydata;
        int tempdata;



        public Form1()
        {
            //Initializations for labels and  buttons
            InitializeComponent();
            label5.Text = "Not Connected";
            label7.Text = "Stopped";
            startRobo.Enabled = false;
            pauseRobo.Enabled = false;
            stopRobo.Enabled = false;

        }
        //-------------------------//
        //-----BLUETOOTH CODE-----//
        //-------------------------//

        //RecieveData used to get bytes from Bluetooth COM port and convert
to string//

        private void ReceiveData()
        {
            int getdata = 0;
            bool started = false;
            Random rRand = new Random();
            while (true)
            {

                try
                {
                    //Send line to request data
                    serialOut.WriteLine("");
                    if (started)
                        getdata++;

                    //Get data as byte and convert to string for display
                    string line = serialOut.ReadByte().ToString();

                    //Display data in texbox
                    dataIn.Invoke(new updateText(UpdateText), line);
                    //Convert data to integer for comparison
                    int datanum = Convert.ToInt32(line);

                    //Check for start signal.  If found, initialize data
capture
                    if (datanum == 255)
                    {
                        dataIn.Invoke(new updateText(UpdateText),
"startbit");
                        started = true;
                    }

                    //Since the signal is splite for each byte the program
loops three times to get the x,y, and temperature
                    if (getdata == 1)
                    {
                        xdata = datanum * 6;
                        dataIn.Invoke(new updateText(UpdateText), "x: " +
xdata);
                    }
                    if (getdata == 2)
                    {
                        ydata = datanum * 6;
                        dataIn.Invoke(new updateText(UpdateText), "y: " +
ydata);
                    }

                    //After getting all three data points, add the to the
heat point structure for graphing
                    if (getdata == 3)
                    {
                        tempdata = datanum;
                        //tempdata = rRand.Next(0, 100);
                        HeatPoints.Add(new HeatPoint(xdata, ydata,
tempdata));
                        getdata = 0;
                        started = false;
                        dataIn.Invoke(new updateText(UpdateText), "t: " +
tempdata);
                    }
                }
                catch
                {

                }
            }
            if (closeRequested)
                closeMe();
            if (disconnectRequested)
                this.Invoke(new disconnectDel(onDisconnect));
        }

        //When the program is disconnected close ports and disable controls

        private void onDisconnect()
        {
            //serialIn.Close();
            serialOut.Close();
            COMout.Enabled = true;
            COMin.Enabled = true;
            stopRobo.Enabled = false;
            startRobo.Enabled = false;
            pauseRobo.Enabled = false;
            connect.Enabled = true;
            label5.Text = "Disconected.";
            threads--;
        }

        //Function to update textbox to show that data was recieved
        //Necessary since the date is drawn from a seperate thread

        private void UpdateText(string s)
        {
            dataIn.Text += "Data In:" + s + "\r\n";
            dataIn.SelectionStart = dataIn.Text.Length;
            dataIn.ScrollToCaret();
            dataIn.Refresh();
        }

        //On close, stop all threads

        private void closeMe()
        {
            threads--;
            this.Invoke(new closeDel(this.Close));
        }

        //When the Connect Button is pressed, setup and connect to chose COM
ports

        private void connect_Click(object sender, EventArgs e)
        {
            string COMInPort = COMin.Text;
            string COMOutPort = COMout.Text;
            //serialIn = new SerialPort(COMInPort, 9600, Parity.None, 8,
StopBits.One);
            serialOut = new SerialPort(COMOutPort, 9600);
            serialOut.ReadTimeout = 1000;
            //serialIn.ReadTimeout = 1000;
            try
            {
                //if (!serialIn.IsOpen)
                //{
                 //   label5.Text = "Opening Inbound Port...";
                  //  serialIn.Open();
                //}
                if (!serialOut.IsOpen)
                {
                    label5.Text = "Opening Outbound Port...";
                    serialOut.Open();
                }
                label5.Text = "Ports Opened, Opening new thread for
listening.";
```

```
                receiveThread = new Thread(new ThreadStart(ReceiveData));
//Create new thread
                threads++;
                receiveThread.Start();
                label5.Text = "Connected.";  //Start new thread

                //Enable/Disable necessary and unnecessary controls

                COMin.Enabled = false;
                COMout.Enabled = false;
                startRobo.Enabled = true;
                pauseRobo.Enabled = true;
                stopRobo.Enabled = true;
                connect.Enabled = false;
            }

            //Return error in message box

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        //Send Start command to robot
        private void startRobo_Click(object sender, EventArgs e)
        {
            try
            {
                serialOut.WriteLine("S");  //Send S for start command
                dataIn.Text += "Out: " + "Start Command" + "\r\n";  //show
that the command was sent in the text bos

                //Force text box to auto scroll
                dataIn.SelectionStart = dataIn.Text.Length;
                dataIn.ScrollToCaret();
                dataIn.Refresh();
            }

            //Return error if message cannot be sent
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        //See startRobo
        private void pauseRobo_Click(object sender, EventArgs e)
        {
            try
            {
                serialOut.WriteLine("P");
                dataIn.Text += "Out: " + "Pause Command" + "\r\n";
                dataIn.SelectionStart = dataIn.Text.Length;
                dataIn.ScrollToCaret();
                dataIn.Refresh();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        //See startRobo
        private void stopRobo_Click(object sender, EventArgs e)
        {
            try
            {
                serialOut.WriteLine("H");
                dataIn.Text += "Out: " + "Stop Command" + "\r\n";
                dataIn.SelectionStart = dataIn.Text.Length;
                dataIn.ScrollToCaret();
                dataIn.Refresh();
                serialOut.Close();
                connect.Enabled = true;
                stopRobo.Enabled = false;
                startRobo.Enabled = false;
                pauseRobo.Enabled = false;
                label5.Text = "Disconnected";

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        //---------------------------//
        //-----HEAT MAP CREATION-----//
        // Adapted from Dylan Vester //
        //---------------------------//

        private Bitmap CreateIntensityMask(Bitmap mapSurface, List<HeatPoint>
aHeatPoints)
        {
            // Initialize new surface to graph on
            Graphics DrawSurface = Graphics.FromImage(mapSurface);

            // Initialize surface as white
            DrawSurface.Clear(Color.White);

            // Takes each heat point from the structure and draws it on the
surface with a given radius
            foreach (HeatPoint DataPoint in aHeatPoints)
            {
                DrawHeatPoint(DrawSurface, DataPoint, 30);
            }

            return mapSurface;
        }

        //Takes data from above and initiates how to plot the data and how
adjust the gradient for the intensity
        private void DrawHeatPoint(Graphics Canvas, HeatPoint HeatPoint, int
Radius)
        {
            List<Point> CircumferencePointsList = new List<Point>();

            Point CircumferencePoint;

            Point[] CircumferencePointsArray;

            // Scale intensity to fit the data
            float fRatio = 1F / 50;
            int iIntensity = (HeatPoint.Intensity - ((HeatPoint.Intensity -
50) * 2));
            float fIntensity = iIntensity * fRatio;

            //Draw the gradient point as a circle by changing the angle and
drawing each part
            for (double i = 0; i <= 360; i += 10)
            {
                CircumferencePoint = new Point();

                CircumferencePoint.X = Convert.ToInt32(HeatPoint.X + Radius *
Math.Cos(ConvertDegreesToRadians(i)));
                CircumferencePoint.Y = Convert.ToInt32(HeatPoint.Y + Radius *
Math.Sin(ConvertDegreesToRadians(i)));

                CircumferencePointsList.Add(CircumferencePoint);
            }

            CircumferencePointsArray = CircumferencePointsList.ToArray();

            //Set gradient brush in order to draw the final gradient
            PathGradientBrush GradientShaper = new
PathGradientBrush(CircumferencePointsArray);

            ColorBlend GradientSpecifications = new ColorBlend(3);

            GradientSpecifications.Positions = new float[3] { 0, fIntensity,
1 };
            GradientSpecifications.Colors = new Color[3]
        {
            Color.FromArgb(0, Color.White),
            Color.FromArgb(HeatPoint.Intensity, Color.Black),
            Color.FromArgb(HeatPoint.Intensity, Color.Black)
        };

            // Pass off color blend to PathGradientBrush to instruct it how
to generate the gradient
            GradientShaper.InterpolationColors = GradientSpecifications;

            // Draw polygon (circle) using our point array and gradient brush
            Canvas.FillPolygon(GradientShaper, CircumferencePointsArray);
        }

        private double ConvertDegreesToRadians(double degrees)
        {
            double radians = (Math.PI / 180) * degrees;
            return (radians);
        }
        //Since the map is created using grayscal gradient, colorize this in
order to better visualize
        public static Bitmap Colorize(Bitmap Mask, byte Alpha)
        {
            //Initialize a new bitmap for the image to be applied to
            Bitmap Output = new Bitmap(Mask.Width, Mask.Height,
PixelFormat.Format32bppArgb);

            Graphics Surface = Graphics.FromImage(Output);
            Surface.Clear(Color.Transparent);

            ColorMap[] Colors = CreatePaletteIndex(Alpha);

            ImageAttributes Remapper = new ImageAttributes();
            Remapper.SetRemapTable(Colors);

            //Apply the surface to the bitmap
            Surface.DrawImage(Mask, new Rectangle(0, 0, Mask.Width,
Mask.Height), 0, 0, Mask.Width, Mask.Height, GraphicsUnit.Pixel, Remapper);

            return Output;
        }

        private static ColorMap[] CreatePaletteIndex(byte Alpha)
        {
            ColorMap[] OutputMap = new ColorMap[256];

            //Utilize a bitmap for applying the color, easily adjustable
            Bitmap Palette = (Bitmap)Bitmap.FromFile(@"C:\Documents and
Settings\Administrator\Desktop\Heat\colors2.bmp");

            //Replace all grayscale colors with corresponding colors
            for (int X = 0; X <= 255; X++)
            {
                OutputMap[X] = new ColorMap();
                OutputMap[X].OldColor = Color.FromArgb(X, X, X);
                OutputMap[X].NewColor = Color.FromArgb(Alpha,
Palette.GetPixel(X, 0));
            }

            return OutputMap;
        }

        //On button click, generate the map and display it
        private void button3_Click_1(object sender, EventArgs e)
        {
            Bitmap heatMapImage = new Bitmap(pictureBox3.Width,
pictureBox3.Height);

            heatMapImage = CreateIntensityMask(heatMapImage, HeatPoints);

            heatMapImage = Colorize(heatMapImage, 255);

            pictureBox3.Image = heatMapImage;

            //Save final image as a png to be utilized with Google Maps
            heatMapImage.Save(@"C:\wamp\www\HeatMap\1.png", ImageFormat.Png);
            button4.Enabled = true;
        }

        //Enable second window to show Google Maps integration
        private void button4_Click(object sender, EventArgs e)
        {
            Form2 Form2 = new Form2();
            Form2.Show();

        }






    }

    //Set structure for storing HeatPoints
    public struct HeatPoint
    {
        public int X;
        public int Y;
        public int Intensity;
        public HeatPoint(int iX, int iY, int bIntensity)
        {
            X = iX;
            Y = iY;
            Intensity = bIntensity;
        }
    }
}
```

Pictures

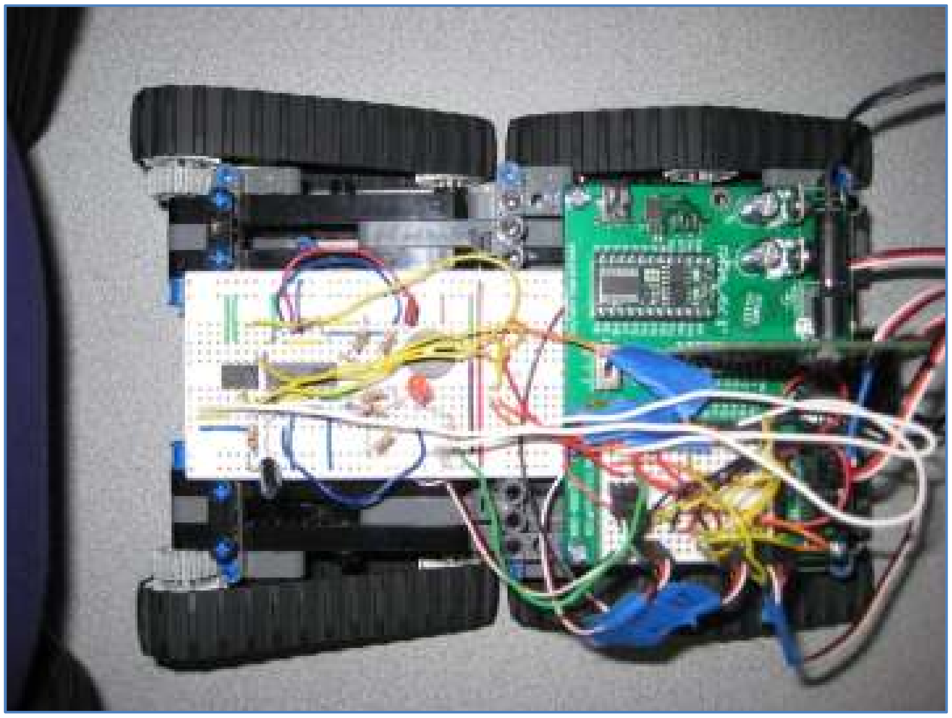


Figure 19: Top View of ThermDetector

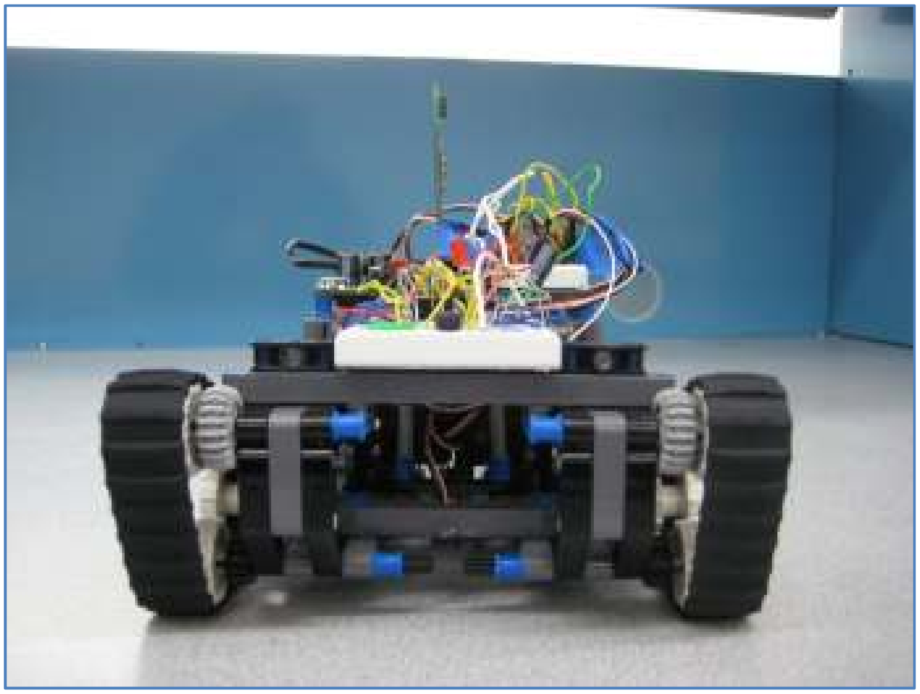


Figure 20: Front View of ThermDetector

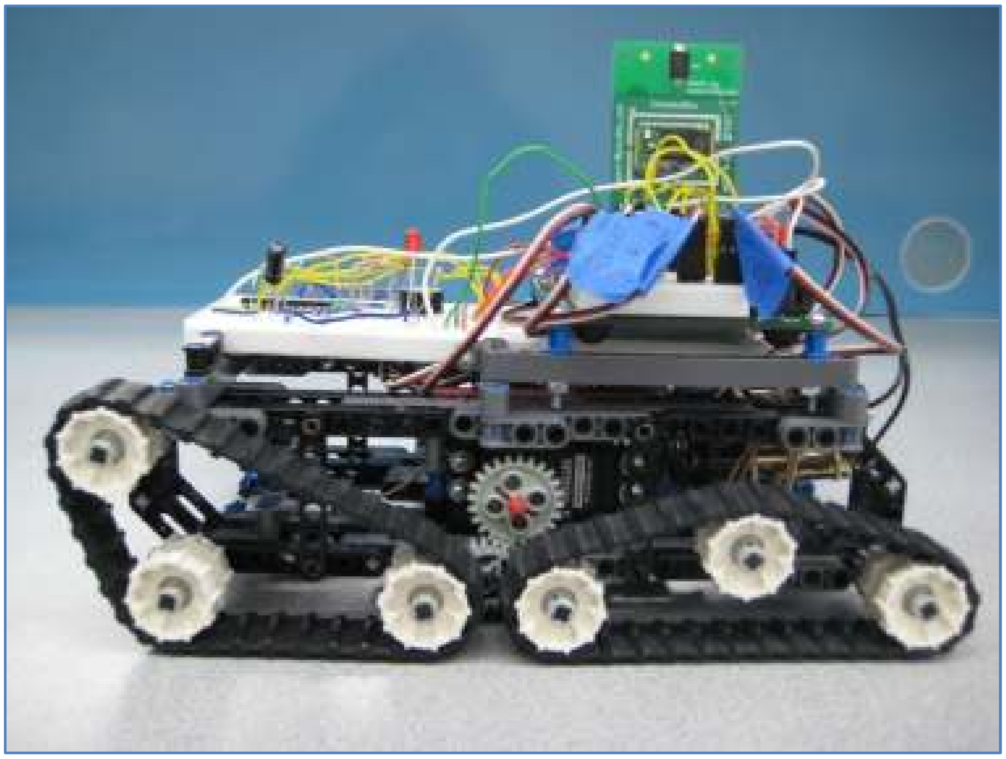


Figure 21: Side View of ThermDetector

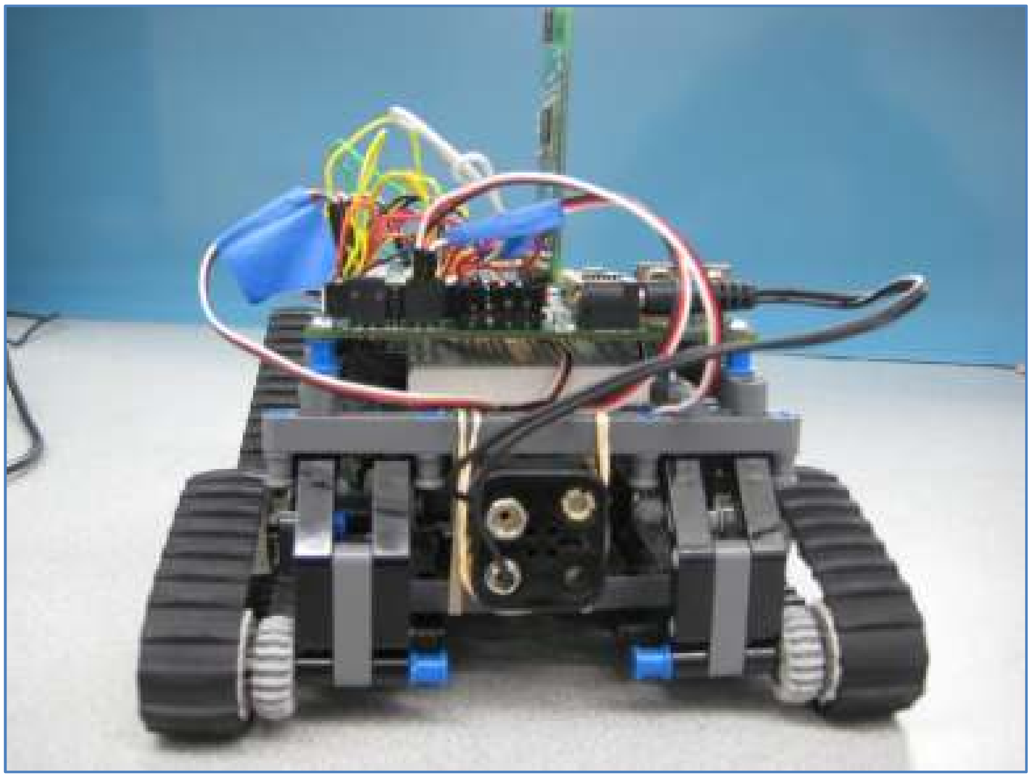


Figure 22: Back View of the ThermDetector

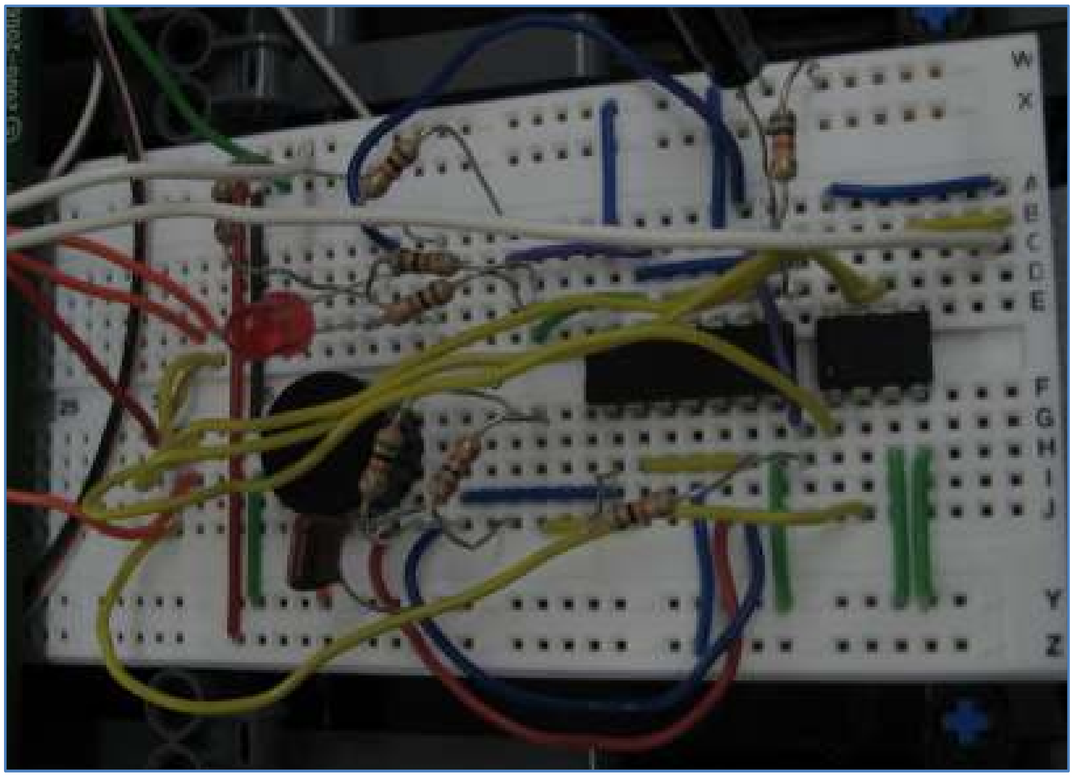


Figure 23: Close-up of Warning system and Hall-effect Sensor Circuits

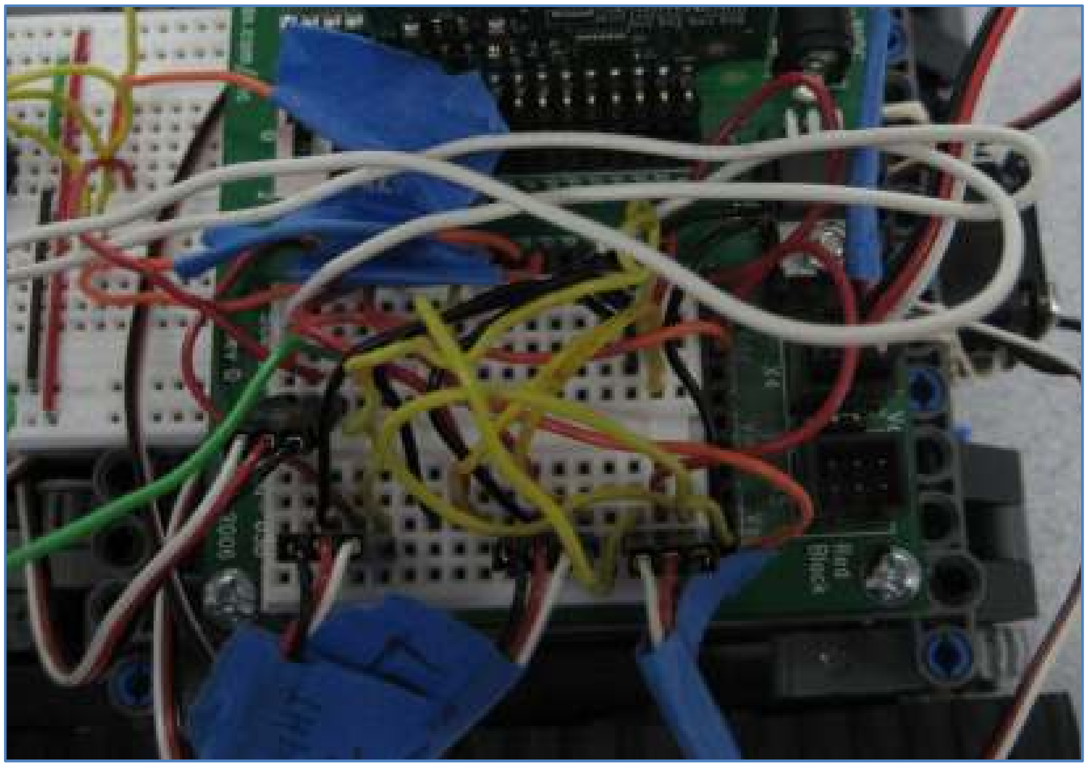


Figure 24: Close-up of Servomotor, Light Sensor, and Temperature Sensor Connections

References

1. www.Parallax.com
    a. Used for obtaining descriptions about various parts
2. www.eoearth.org/article/coalfire_and_remote_sensing
    a. Provided information concerning coal mine fires
3. www.wikipedia.org/wiki/coal_mine_fire
    a. Useful information concerning coal mine fires.
4. ME5643: Mechatronics Lectures
    a. Useful for basic circuit diagrams, and theory.
5. Parallax Basic Stamp Books by: Andy Lindsay
    a. What’s a Microcontroller?
    b. Basic Analog and Digital
    c. Robotics with the Boe-Bot
    d. Smart Sensors & Applications.
6. www.coolmagnetman.com/magmeter.htm
    a. Basis for building Gaussmeter.
7. http://dylanvester.com/post/Creating-Heat-Maps-with-NET-20-(C-Sharp).aspx
    a. For heat map development
8. www.codeproject.com
    a. For serial Bluetooth assistance