MINDANAO STATE UNIVERSITY - ILIGAN INSTITUTE OF TECHNOLOGY

COLLEGE OF ENGINEERING

DEPARTMENT OF EECCE

**SEMI-AUTONOMOUS PATH FINDING ROBOT**

PROJECT “BLACKY”

A Requirement for EE 152

**Industrial Electronics and Instrumentation**

**SUBMITTED TO**

**ENGINEER JABIAN, MARVEN E.**

DEPARTMENT ELECTRICAL/E.C.E./COMPUTER ENGINEERING

COLLEGE OF ENGINEERING

**PROPONETS**

ABESTANO, JOHANNAH MAE

AMIL, CHRISTIAN NICOLE

AQUINO, JOHANNA JEAN

ARDA, MARGIE SAAVEDRA

BATE, MARNIER

CALIBAYAN, SHARYL NAVARRO

ENANOR, CARYL KEEN

HADJI ABDULLA, FAI HANIMA

PALLER, KIM

RATERTA, ANTHONY

REGALADO, GIL MICHAEL

VISITACION, STEPHANIE REGIS

**INTRODUCTION**

Today, the development and familiarity of the general population on autonomous vehicle technology is prevalent. In this study, the researchers will try to develop an autonomous vehicle that operates in small scale. Using only off the shelf technology that are affordable and practical.

Blacky is a semi-autonomous direction aware and collision avoiding robot. Blacky has 3 Ultrasonic Distance Sensors, 1 Digital Compass and 2 DC Motors. It is on a E-Gizmo PBOT Chassis, and is powered by a 6 AA Batteries and a separate 9V Battery for the onboard PIC18F4550 Microcontroller.

Blacky can communicate wirelessly for 500 Meters from the base station and can transmit data at 9600 baud rate using JZ863 RF Module. Blacky also has output RX/TX that can be wired directly for the debugging devices that may be attached to it. Furthermore Blacky has an Input Port for the PICKIT 3 Programmer from Microchip.

At initialization, Blacky will save the current direction with respect to the true North. When this is saved, Blacky will begin moving forward. If one of the 3 Ultrasonic Devices detect objects that block Blacky's path, Blacky will turn left or right depending on which of the two sensors have the furthest unblocked path. If both sensors does not register a blockage, the Blacky will take Right.

If Blacky takes the right direction, the left sensor will continue to scan if the Blocking object is still in view, if not, Blacky will return to the original direction as saved in the beginning. Blacky can control its speed depending on the distance of the Blocking Object. If the blocking object is far away, then Blacky will move forward faster, otherwise slower.

**OBJECTIVE**

The focus of this project is to develop a semi-autonomous mobile vehicle. The robot must be able to traverse an area safely by avoiding obstacles while keeping the correct direction heading in order not to lose track of the final objective which is to arrive at another point as designated.

1. Robot must avoid obstacles.
2. Resolve correct heading when there are no longer any obstacles in sight.

**METHODOLOGY**

Blacky’s development will undergo the following steps to better choose and design all the related systems that comprise Blacky as a whole.

1. Initial design of a block diagram involving the key hardware and software components of the system.
2. Determination of tools, parts and other associated software and hardware peripherals.
3. Acquisition of such components and testing to determine their usability and effectiveness if ever included in the final design.
4. Design of circuitry, hardware, mechanical and software and firmware peripherals.
5. Assembly electronics and mechanical components and compilation of necessary firmware and software peripherals.
6. Development of effective algorithms to solve Collision Avoidance and Heading Resolution.
7. Testing and fine tuning of algorithms involved in object avoidance, navigation and communication.
8. Final review and debugging of electrical, mechanical and software components.

**1. INITIAL DESIGN PHASE**

In order for the project to be successful in meeting its goals. The following key features were slated for development.

1. Motors and Chassis easily mountable with custom designed systems.
2. Motor control system for navigation.
3. Distance sensors to detect obstacles.
4. A digital compass to keep track of heading and robot orientation.
5. Wireless RF device to send data to a base station.
6. Robot monitoring from the base station.
7. A microcontroller capable of handling all robot based systems including sensors and motor drivers.

A simple block diagram is then developed out of these key requirements.



On the side of the base station, a simple system is also developed.



**2. DETERMINATION OF PARTS**

Considered during the determination of parts where available budget, availability and reliability. After much research and deliberation the following parts were selected for their respective purposes.

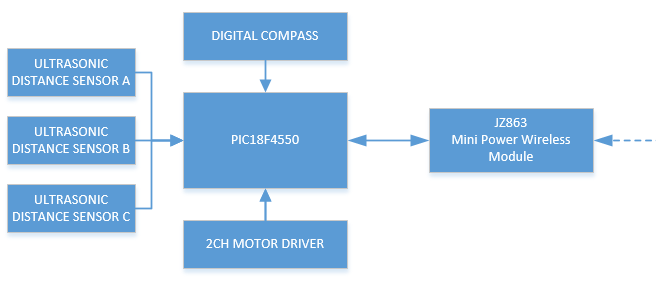
|  |  |
| --- | --- |
| **OBSTACLE SENSORS**  ULTRASONIC DISTANCE SENSOR  It was decided that three of these sensors would be sufficient to monitor front, left and right sides of the robot for the purpose of obstacle detection. | http://www.e-gizmo.com/KIT/images/ultrasonicsonar/ultra%20sonic%20Distanceranging%20sensor.png |
| **DIGITAL COMPASS**  This digital compass was selected for navigation aid for the robot. After avoiding obstacles, the robot is able to review its current heading and correct if necessary to really direct itself to the correct path. | http://www.e-gizmo.com/KIT/images/digital%20compass/digital%20compass.png |
| **MODEL 863 RF WIRELESS DATA TRANSCEIVER**  JZ863, the mini power wireless module, is used as the wireless data transmission in short distance. With the small size, weight and power consumption and good stability and reliability, it has the function of bidirectional data sign transmission, test and control. | http://www.e-gizmo.com/KIT/images/863/863.png |
| **2 CHANNEL DC MOTOR DRIVER WITH SPEED CONTROL**  This solid state motor controller can provide independent control to two DC motors. Functions include FORWARD, REVERSE and STOP.  Furthermore, a 4 bit common speed control port (16 levels) is provided as an added feature. Motor speed control is put in effect by varying the duty cycle of the output drive. | http://www.e-gizmo.com/KIT/images/2channeldcmotordriver/2%20Channel%20DC%20Motor%20Driver%20with%20Speed%20Control.png |
| **CHASSIS BOT 2R0 ENTRY LEVEL MOBILE ROBOT KIT**  This chassis was the chosen standard as decided by the whole class. Only the chassis and two motors were acquired. The picture though includes some associated circuitry but they were not included during purchase. | http://www.e-gizmo.com/KIT/images/PBOT2/pbot.png |
| **MICROCONTROLLER – PIC18F4550**  The chosen microcontroller was the PIC18F4550. It was assumed that it could handle all the interfacing of components and the monitoring with wireless communications. | http://picafio.files.wordpress.com/2010/01/18f4550.jpg |
| **INTERFACE – ARDUINO MEGA 2560**  The Arduino Mega 2560 which was readily available was decided to be the main interface from the transceiver of the receiving end to the computer which as the monitoring software. | https://dlnmh9ip6v2uc.cloudfront.net/images/products/9/9/4/9/09949-01.jpg |
|  |  |

**3. ACQUISISTION OF COMPONENTS**

Much of the major components were acquired from a single vendor which is E-Gizmo Mechatronix Central and the others were acquired from local stores because they are generic parts.

**4. CUSTOM CIRCUIT DESIGN**

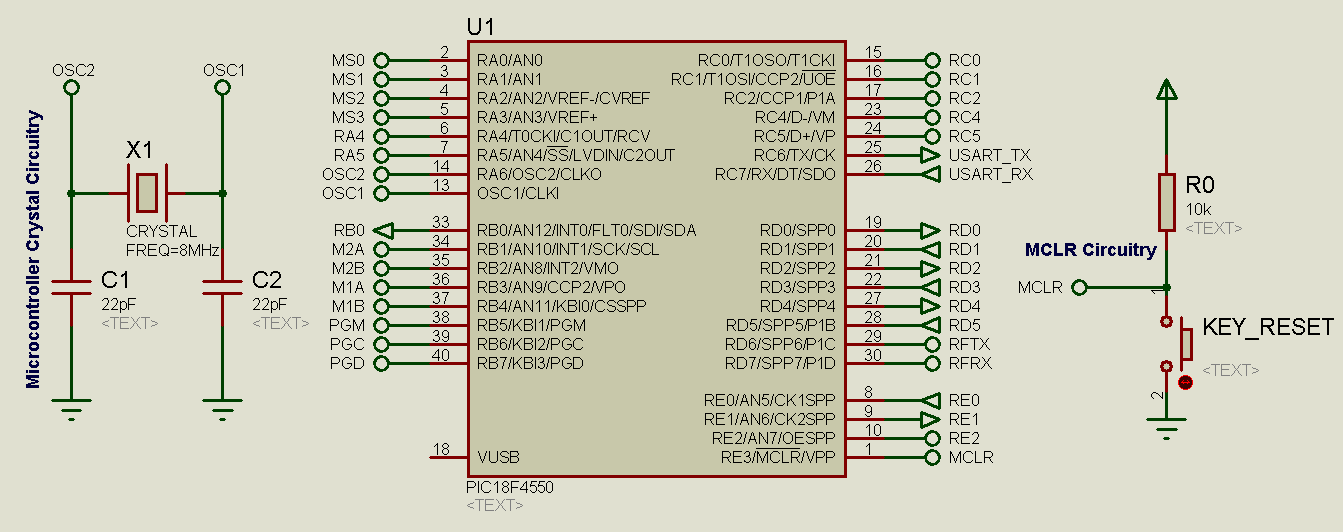
A more detailed Block diagram is developed.





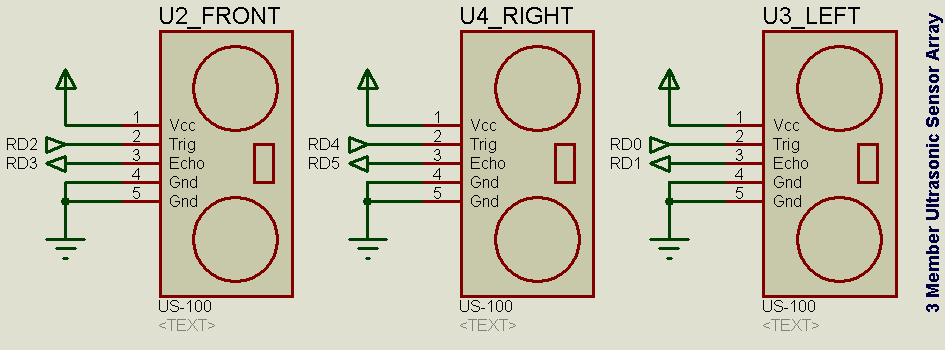
The main challenge for the project was the efficient integration of all sensors and communications components.

**MICROCONTROLLER BLOCK**



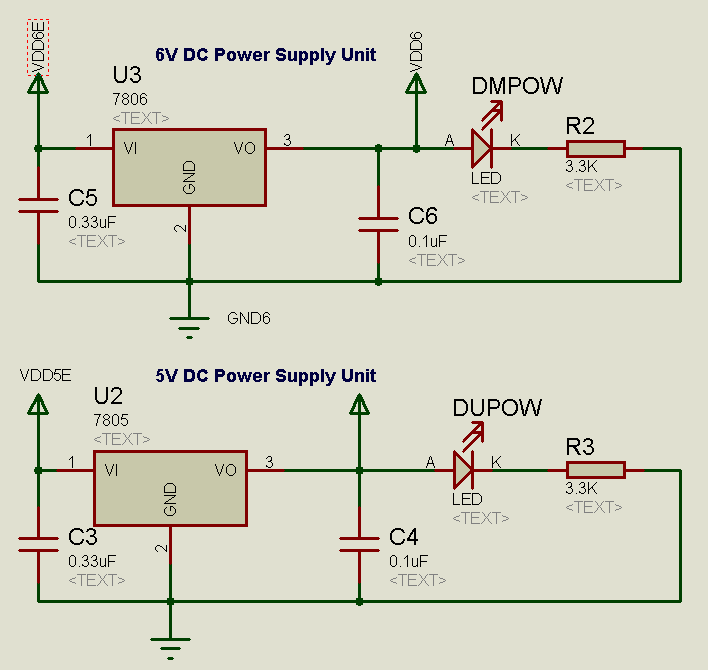
The microcontroller is powered by an 8 MHz Crystal Oscillator. However, internal microcontroller PLL subsystem allows it to operate at 48 MHz providing us with decent speeds.

**SENSOR ARRAY**



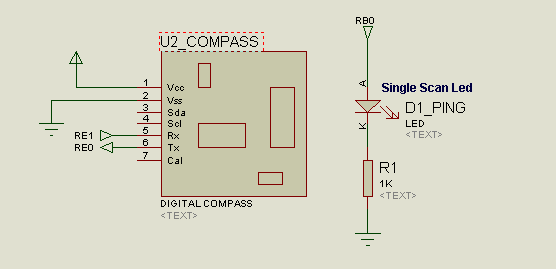
The three Ultrasonics Distance Sensor have connections to the main microcontroller at RD0:RD5.

**POWER SUPPLY BLOCK**



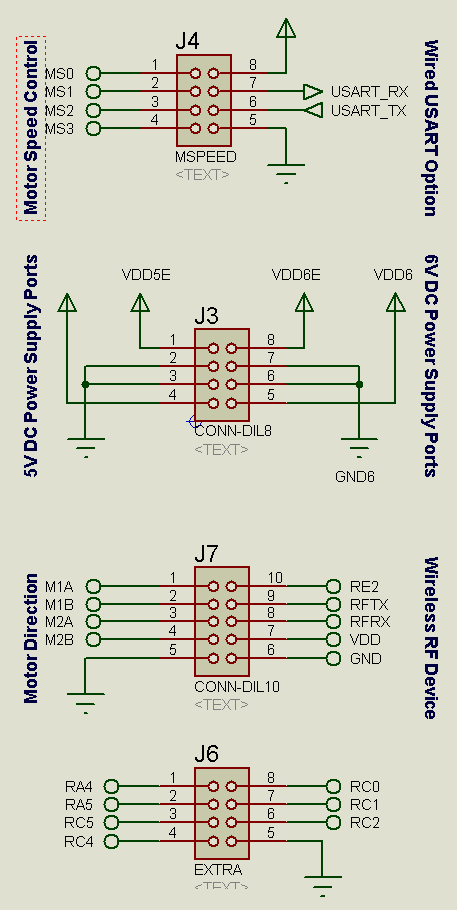
The initial design had included a 5V supply for the 5V logic circuitry and 6V for the motors. However, during the final assembly, the 6V supply was never used and motors were directly connected to an array of six AA Batteries.

**DIGITAL COMPASS BLOCK**



The digital compass was connected in the following manner. And a single scan led is attached to RB0 to show the user if the sensors are scanning and the transceiver is sending and receiving data from the base station correctly. The single scan led should blink after each roundtrip of data from base station to robot.

**OTHER PERIPHERALS**



Motor Speed – MS0:MS3 for 16 Level Speed Control to the 2 CH Motor Driver

Wired USART – Mainly used for debugging purposes.

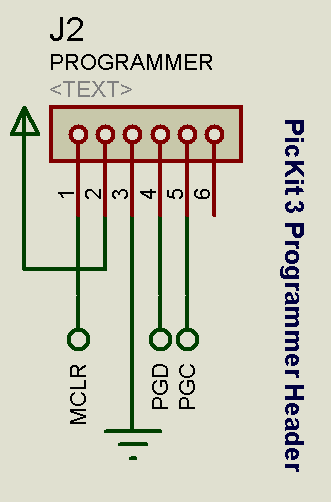
5V/6V DC Supply Ports – Input and Output ports for external Peripherals.

Motor Direction Control – M1A:M2B controls the 2CH Motor Driver

Wireless RF Device – Sleep, TX, RX and Supply of the Wireless RF Device.

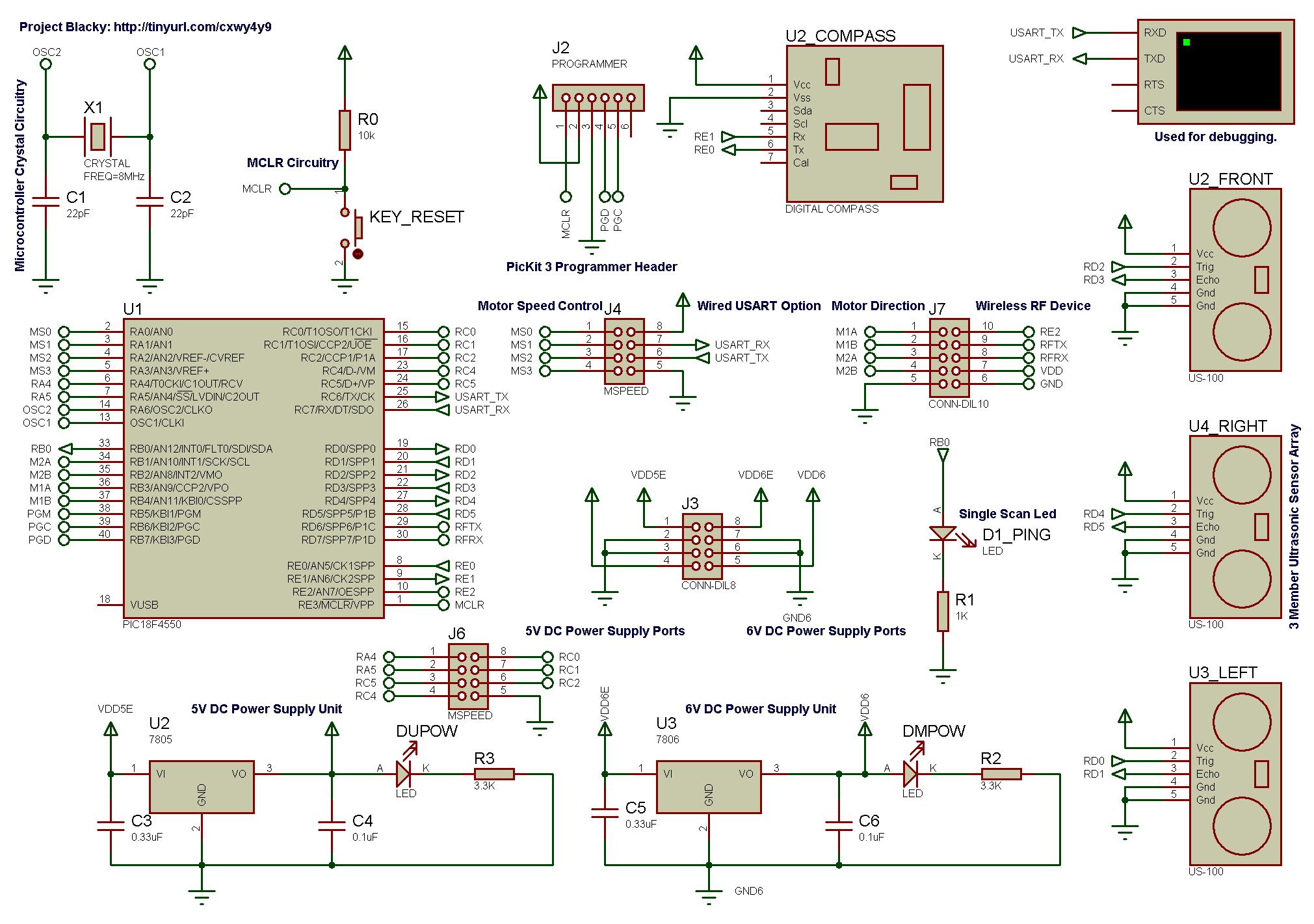
Extra Ports – For other add-ons to the robot.

**ICSP – IN-CIRCUIT SERIAL PROGRAMMER HEADER PINS**



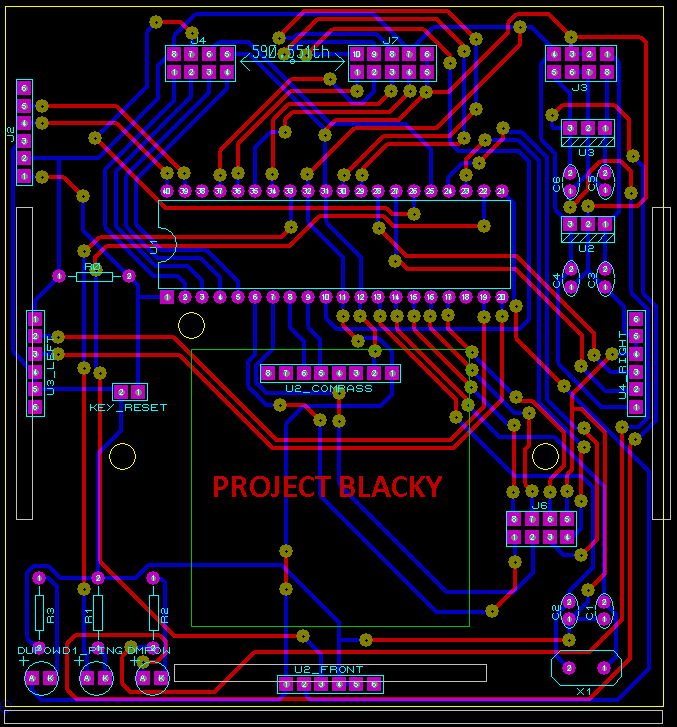
This is included for easy re-programmability of the PIC18F4550.

**OVERVIEW**

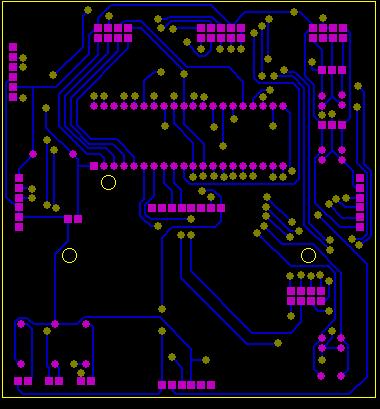


**PCB DESIGN**

The next step is to design a PCB. However one of the few difficulties were the size restrictions of the design and the limited placement of components. It was decided to keep the final design as modular as possible. This required us to resort to a double sided PCB solution.

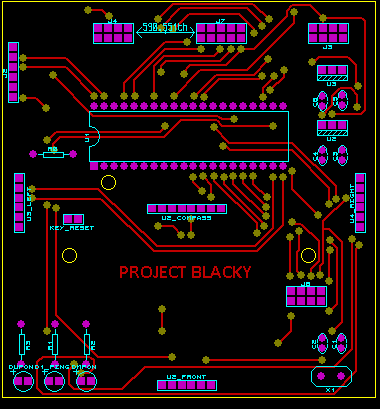


**PCB BACK SIDE**



Much of the routing that are vertical are put in to the back side. Circuitry that are difficult to place are then ported out to the front side using VIAs at strategic locations.

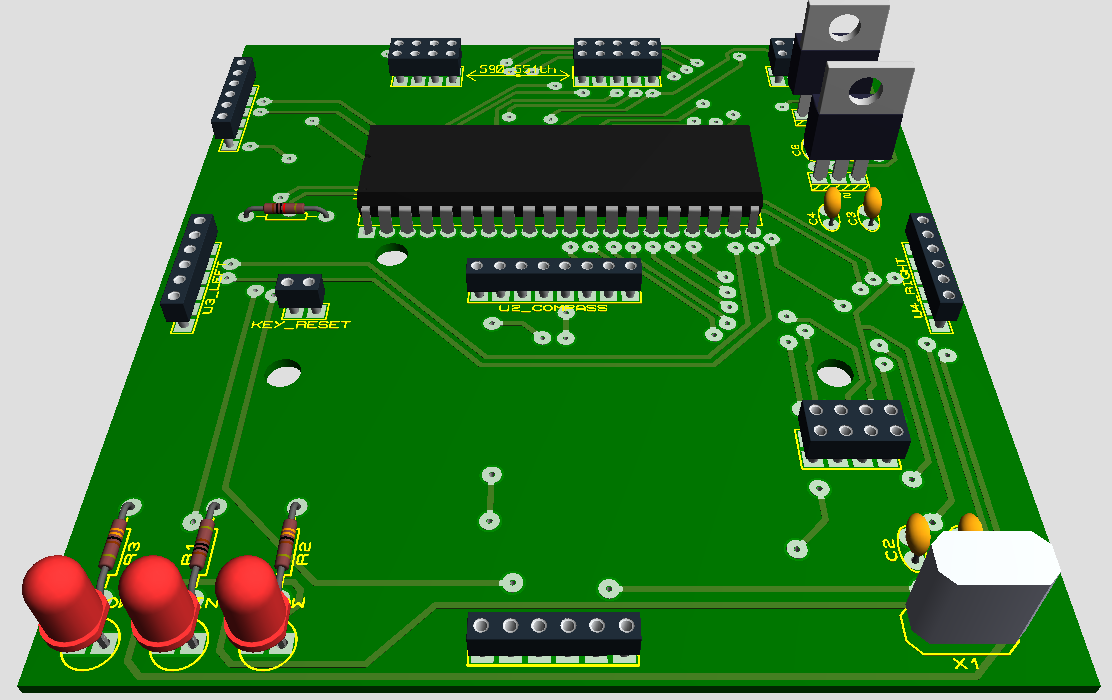
**PCB FRONT SIDE**

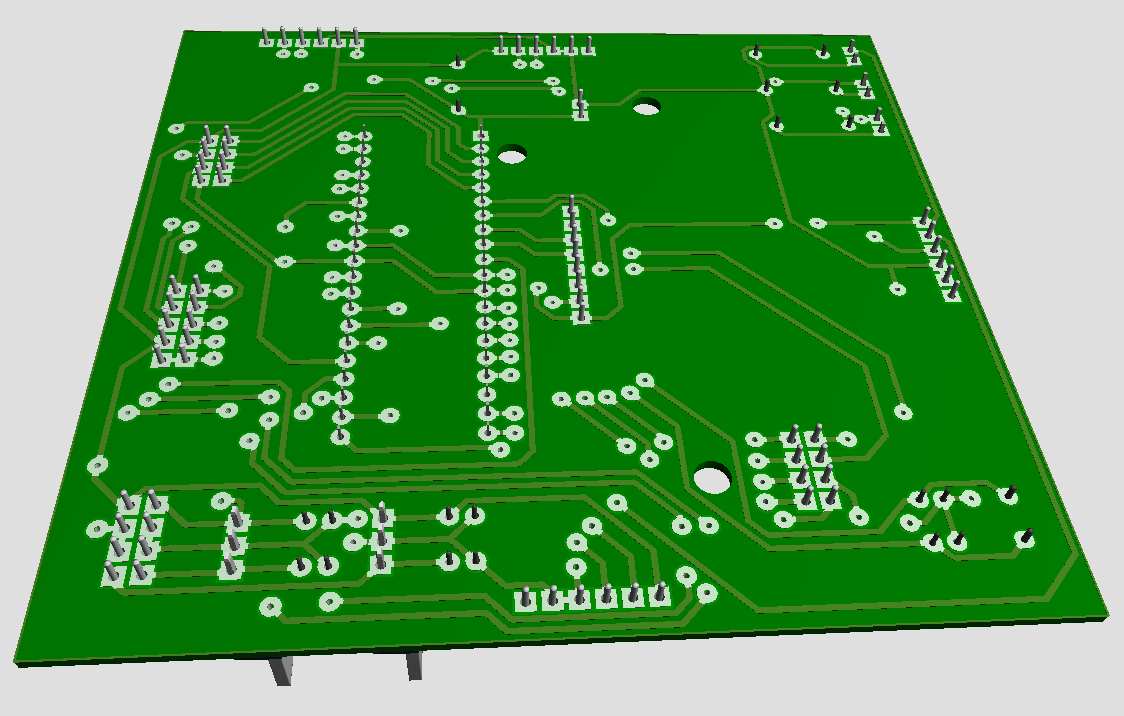


As you can observe much of the routing that are horizontal are placed on the front side. Also through hole pins were generally not connected to VIAs directly. They had to go from the back side, then VIAs, then to the front. This was avoided to simplify the soldering process.

One of the key improvements that can be made to this design is the placement of the Crystal X1 which is very far from the microcontroller. This was a mistake and should have been avoided in the first place.

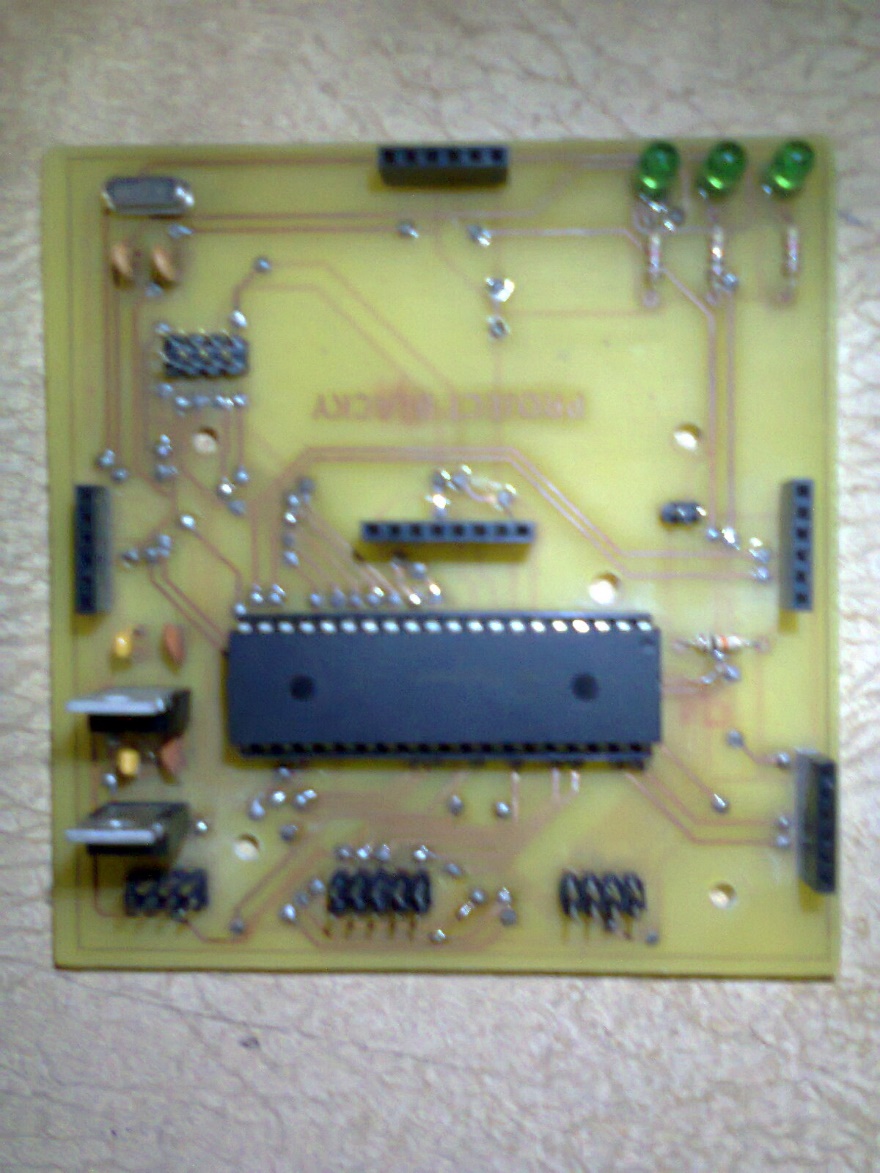
**PREPRODUCTION CHECK**

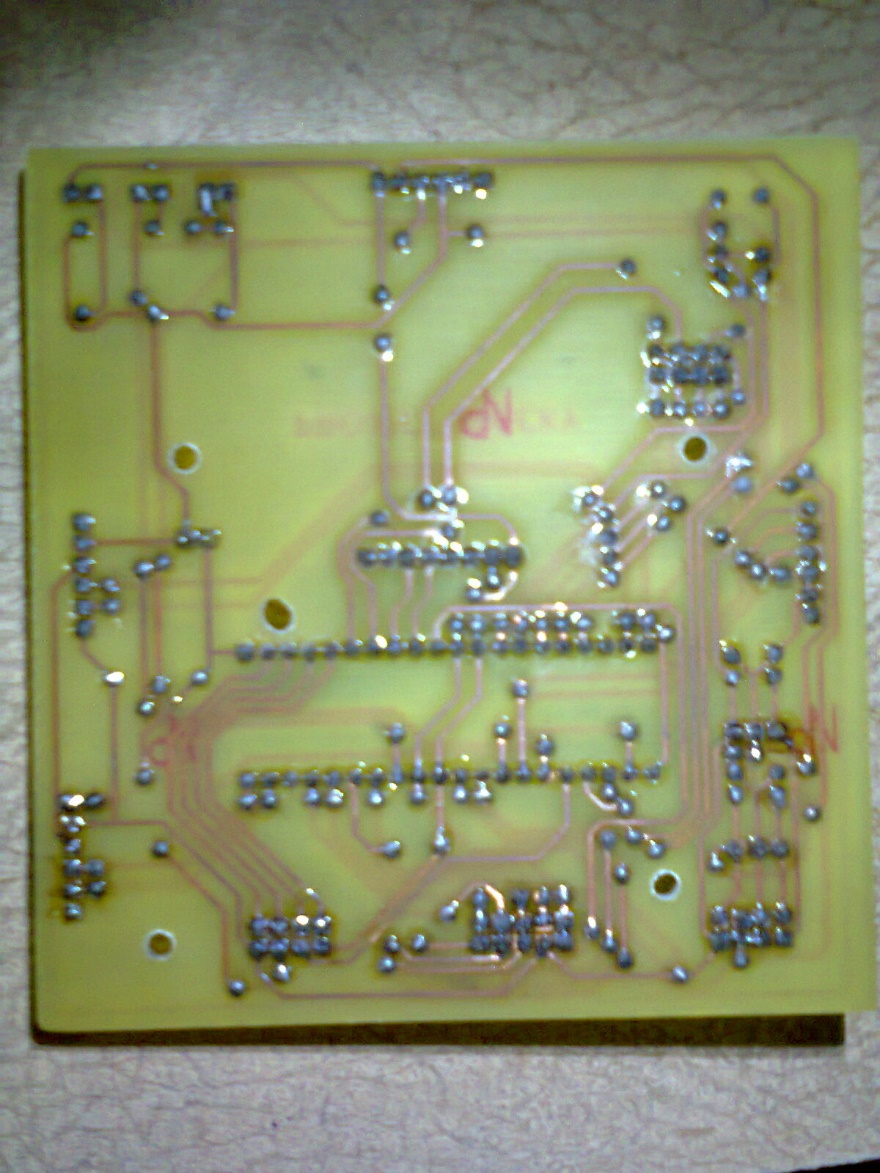




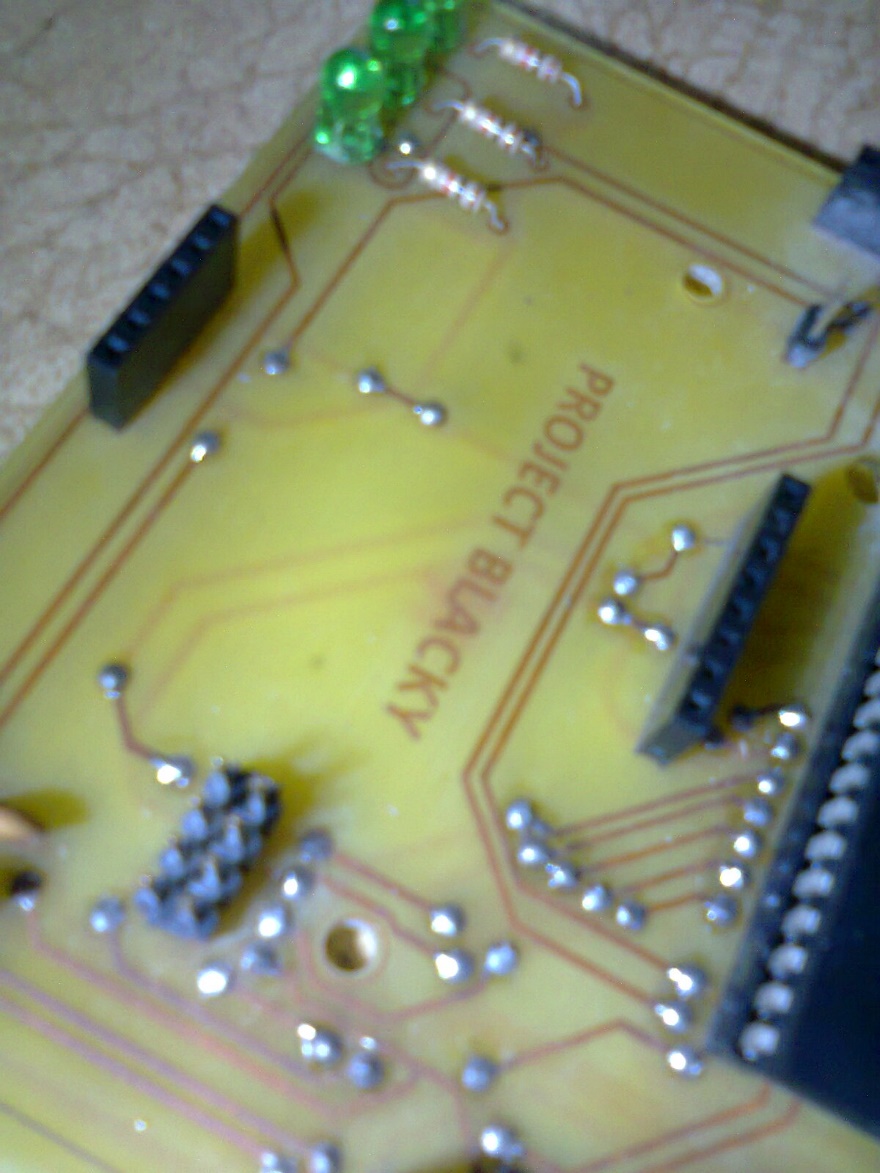
**5. ASSEMBLY OF COMPONENTS**

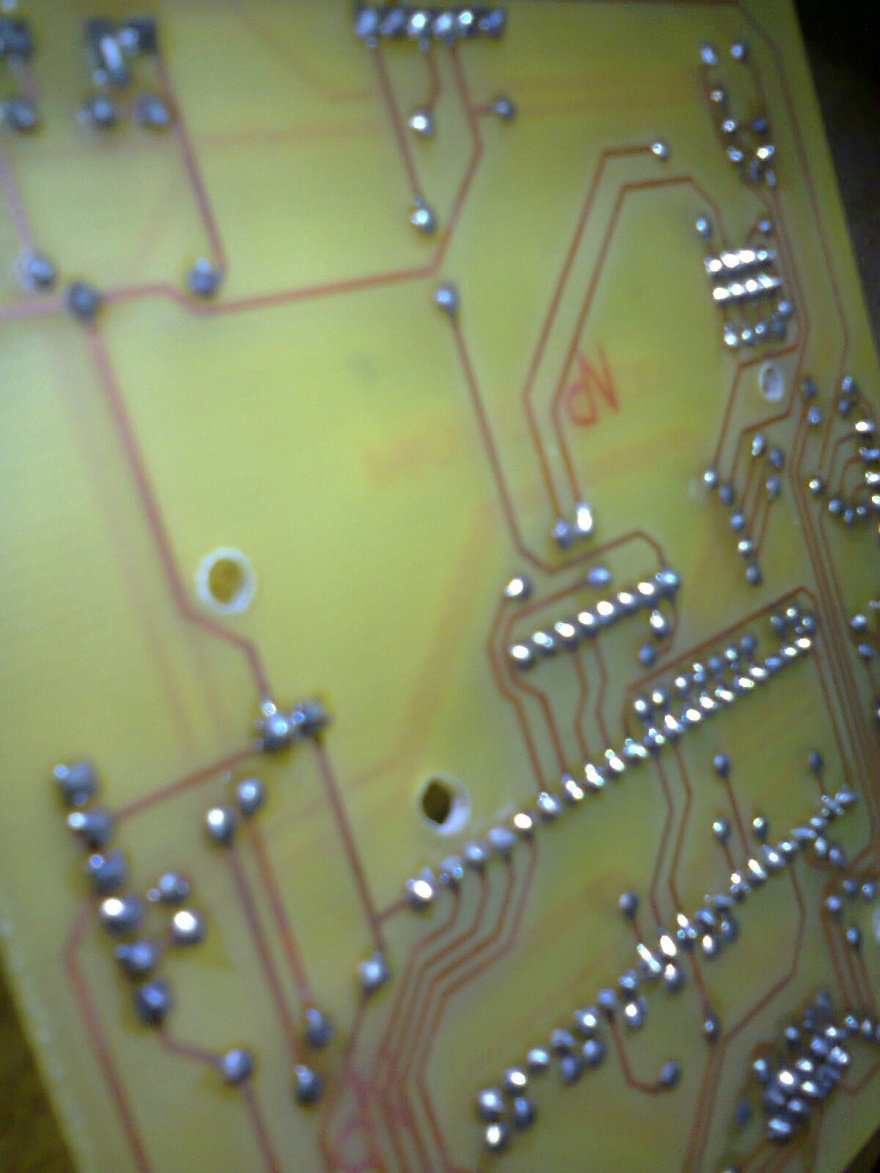
**AFTER ETCHING AND SOLDERING OF MAIN PCB BOARD.**



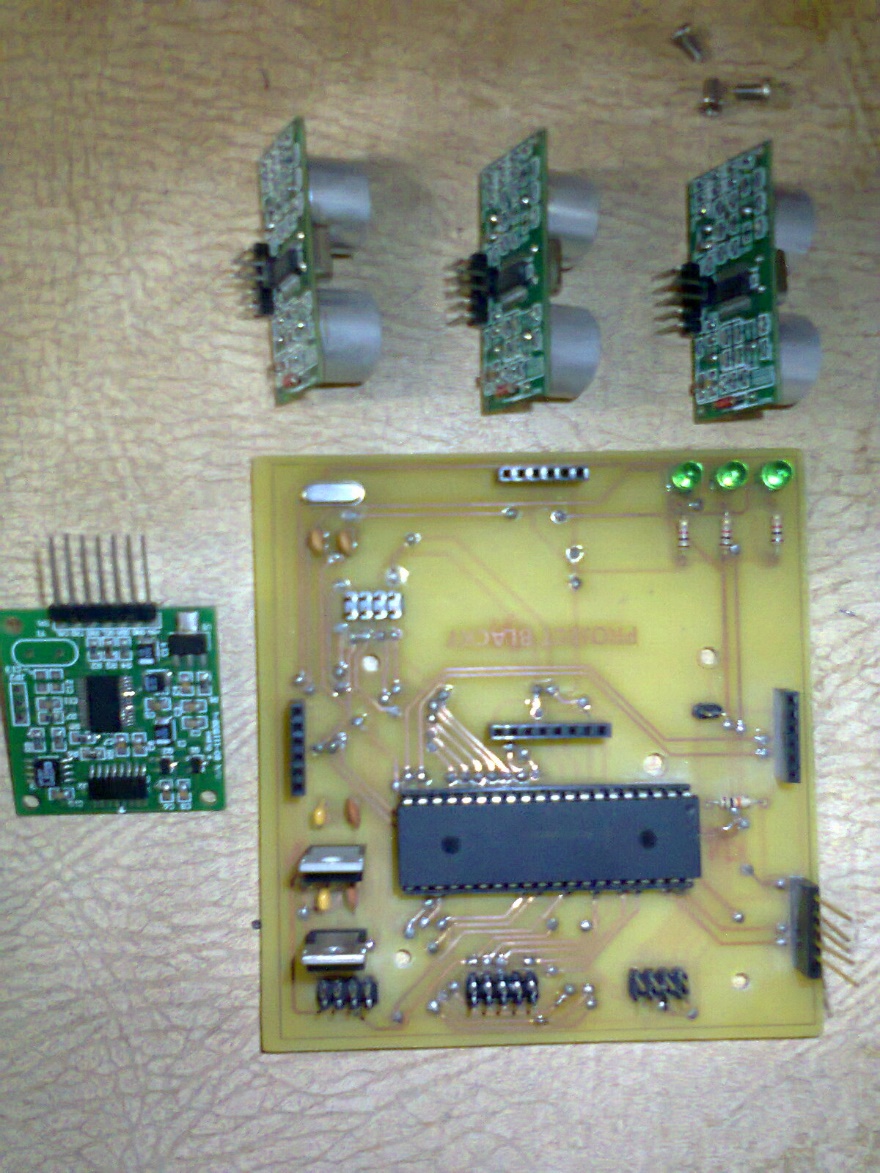


**CLOSE-UP VIEWS OF THE BOARD**



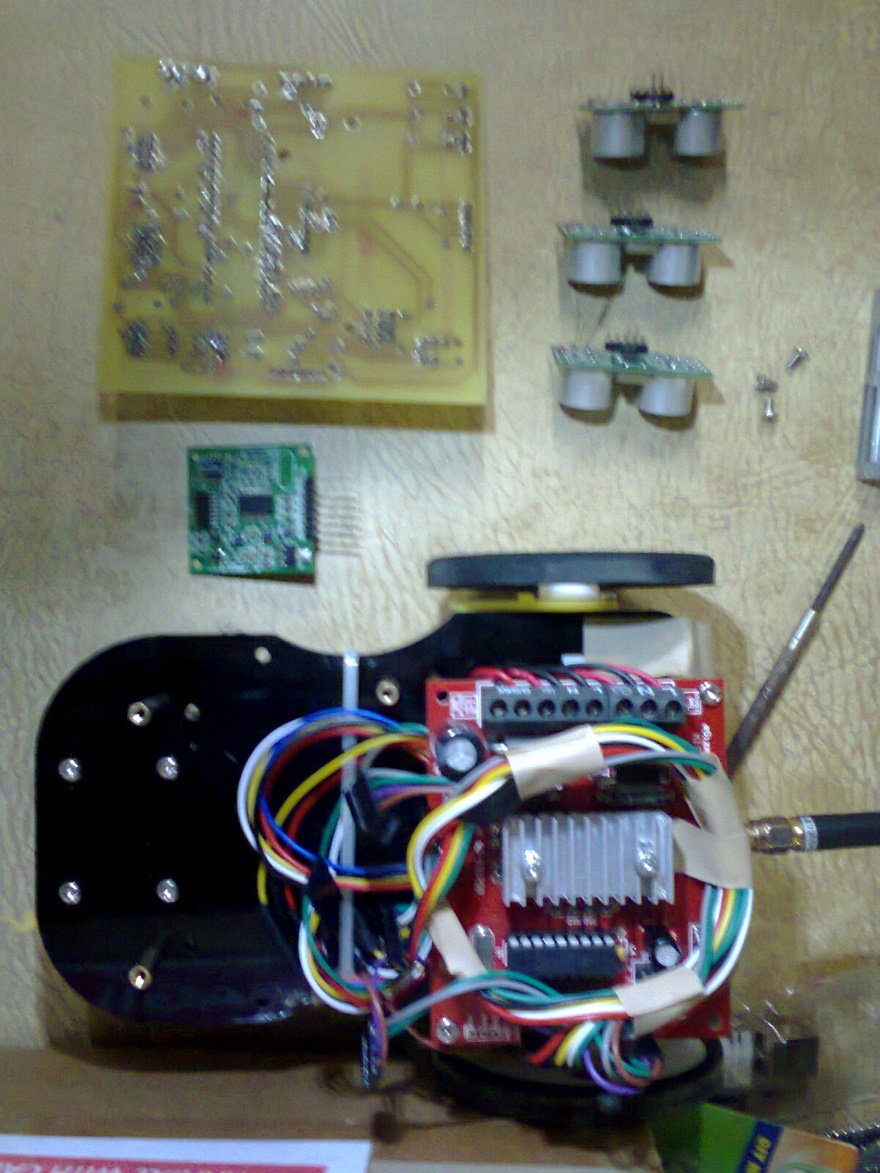


**MAIN BOARD AND ITS SUBCOMPOENTS**





The 2CH Motor Driver and the Wireless RF Device seen in the left already attached to the Chassis.



2 AA Batteries are stuck beneath the Wireless RF Device. Above the Wirless RF Device iss the 2CH Driver.



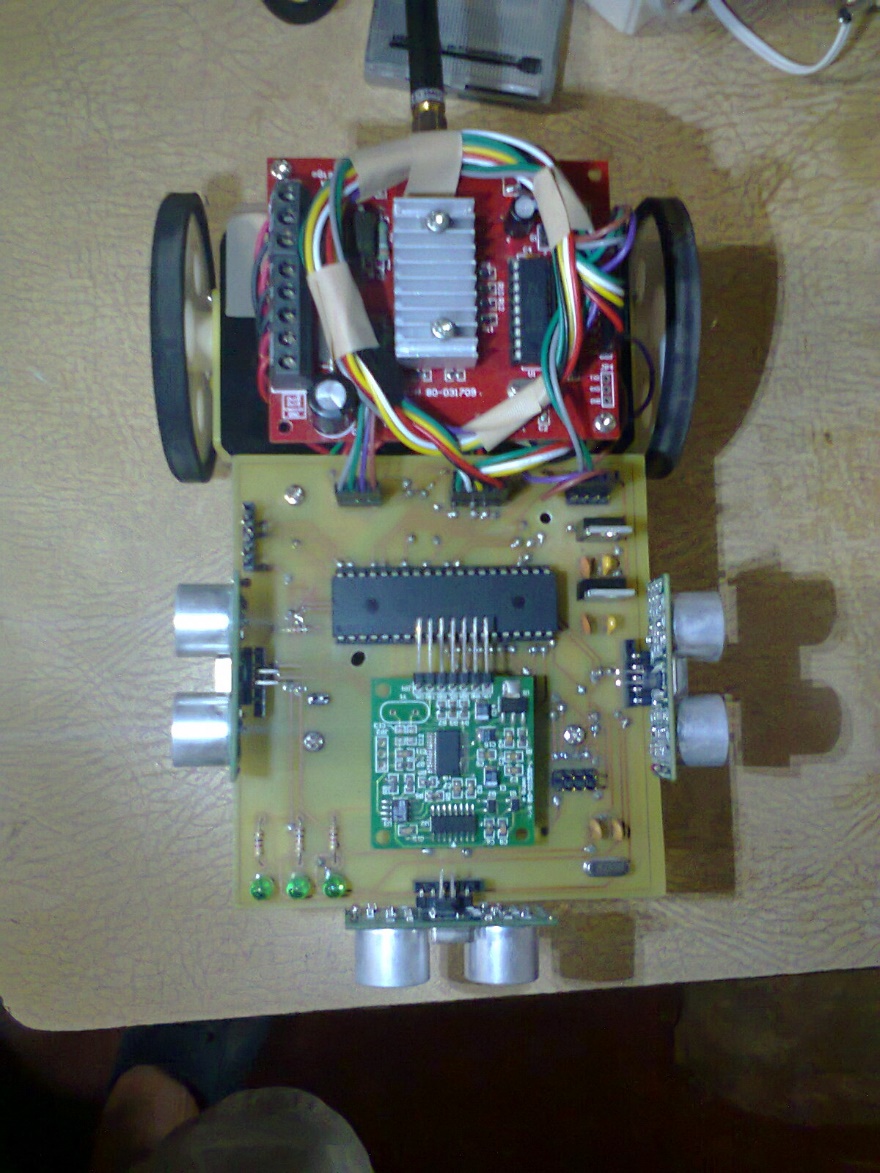
2 More 9V batteries are stuck beneath this taped area. One for the Wireless RF Device and another for the PIC and 2CH Motor Driver.

**PROGRAMMER ACCESS THROUGH ICSP**

****

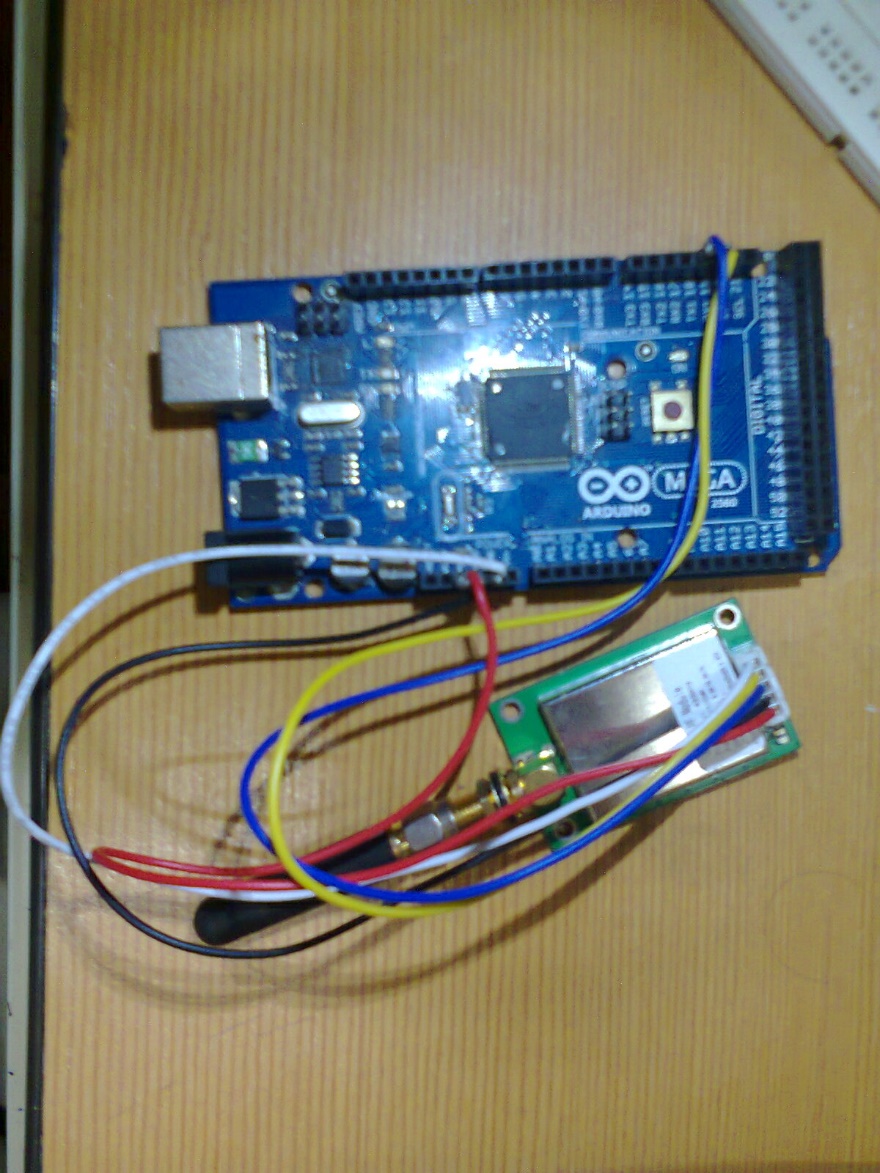


**FINAL ASSEMBLY**





**BASE STATION ASSEMBLY**

****

The USB Port of the Arduino connects to the PC. A valid Com Port is then addressed using the software client in order to monitor Blacky’s performance.

**5. FIRMWAVE AND SOFTWARE DEVELOPMENT & TESTING**

**ALGORITHM**



**FIRMWARE**

Due to the modular setup of the robot, the firmware was also developed in a very modular manner. There will be some code that will not be included here to shorten the paper. There will be an appendix page in the near end of this document detailing where to get resources for the project online through the internet.

|  |
| --- |
| **OBSTACLE DETECTION**  libPing.c – Contains obstacle distance code. This code did not have actual measuring units calibrated. |
| #include "libPing.h"  unsigned int ReadPing\_L**()** **{**  long count\_distance **=** 0**;**  TRISD **=** 0x6A**;**    //UART1\_Write(0x41); // signal A    Trigger\_L **=** 0x01**;**  Delay\_us**(**20**);** // Delay 20us to wait for rising edge.  Trigger\_L **=** 0x00**;**  **while** **(**Echo\_L **==** 0**)**  **{**  //UART1\_Write(0x42); // signal B  **}**  **while** **(**Echo\_L **==** 1**)**  **{**  count\_distance**++;**  //UART1\_Write(0x43); // signal C  **}**  **return** Count\_Distance**((**double**)**count\_distance**,** **(**double**)**MaxDistance**,** **(**double**)**MaxDistanceScaled**);**  **}**  unsigned int ReadPing\_R**()** **{**  long count\_distance **=** 0**;**  TRISD **=** 0x6A**;**    //UART1\_Write(0x45); // signal E    Trigger\_R **=** 0x01**;**  Delay\_us**(**20**);** // Delay 20us to wait for rising edge.  Trigger\_R **=** 0x00**;**  **while** **(**Echo\_R **==** 0**)**  **{**  //UART1\_Write(0x46); // signal F  **}**  **while** **(**Echo\_R **==** 1**)**  **{**  count\_distance**++;**  //UART1\_Write(0x47); // signal E  **}**  **return** Count\_Distance**((**double**)**count\_distance**,** **(**double**)**MaxDistance**,** **(**double**)**MaxDistanceScaled**);**  **}**  unsigned int ReadPing\_F**()** **{**  long count\_distance **=** 0**;**  TRISD **=** 0x6A**;**  //UART1\_Write(0x66);    Trigger\_F **=** 0x01**;**  Delay\_us**(**20**);** // Delay 20us to wait for rising edge.  Trigger\_F **=** 0x00**;**  **while** **(**Echo\_F **==** 0**)**  **{**  //UART1\_Write(0x50); // signal P  **}**  **while** **(**Echo\_F **==** 1**)**  **{**  count\_distance**++;**  //UART1\_Write(0x49); // signal G  **}**    **return** Count\_Distance**((**double**)**count\_distance**,** **(**double**)**MaxDistance**,** **(**double**)**MaxDistanceScaled**);**  **}**  unsigned int Count\_Distance**(**double cnt**,** double max**,** double scale**)** **{**  **if** **(**cnt **>** max**)**  cnt **=** max**;**  **return** **(**unsigned int**)(**scale**\*(**cnt**/**max**));**  **}** |
| Header |
| #ifndef LIBPING\_H  #define LIBPING\_H  #define MaxDistance 2000  #define MaxDistanceScaled 255  extern sfr sbit Echo\_L**;** // Echo bit for the left sensor  extern sfr sbit Echo\_F**;** // Echo bit for the front sensor  extern sfr sbit Echo\_R**;** // Echo bit for the right sensor  extern sfr sbit Trigger\_L**;** // Trigger bit for the left sensor  extern sfr sbit Trigger\_F**;** // Trigger bit for the front sensor  extern sfr sbit Trigger\_R**;** // Trigger bit for the right sensor  unsigned int ReadPing\_L**();**  unsigned int ReadPing\_F**();**  unsigned int ReadPing\_R**();**  unsigned int Count\_Distance**(**double cnt**,** double max**,** double scale**);**  #endif |

|  |
| --- |
| **DIGITAL COMPASS**  libCompass.c – Code responsible for extracting current heading of the robot from the Digital Compass sensor. |
| #include "libBlacky.h"  #include "libCompass.h"  #include "libRF.h"  unsigned char ReadCompass**(**unsigned char send\_serial**,** unsigned char **\*** read\_data**)** **{**  unsigned char buffer**[**12**]** **=** **{**0**};**  unsigned char soft\_uart\_error**;**  unsigned char skewed\_output\_integer**;**  unsigned int out\_buffer\_integer**;**    TRISE **=** 0x09**;**  LATE **=** 0b1001**;**  // Delay\_us(20);    **if** **(**Soft\_UART\_Init**(&**PORTE**,** 0**,** 1**,** 9600**,** 0**)** **==** 0**)** **{**  // Delay\_us(20);    // ingon ani nalang para mo work. kai kung mag social2  // pa ug loop2 murag ma gubot ang Soft\_UART\_Read tungod  // sa extra overhead sa looping mechanism.  buffer**[**0**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**1**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**2**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**3**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**4**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**5**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**6**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**7**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**8**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**9**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**10**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**11**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**  buffer**[**12**]** **=** Soft\_UART\_Read**(**soft\_uart\_error**);**    // after gikuha ang daghan na data samples, ayha pa  // pangitaon kung aha dadto ang actual data. 2x ang sample  // para ma sure2 jud na naay data makuha sa sulod.  Delay\_ms**(**20**);**  out\_buffer\_integer **=** FindDataFromBuffer**(**buffer**,** 12**,** 7**,** read\_data**,** send\_serial**);**  Delay\_ms**(**20**);**    **return** Skew**(**out\_buffer\_integer**,** 359**,** 124**);**  **}** **else** **{**  **if** **(**send\_serial **==** 1**)** **{**  transmit\_rf**(**0xFF**);**  transmit\_rf**(**0xFF**);**  transmit\_rf**(**0xFF**);**  **}**  **}**  **return** 0**;**  **}**  unsigned int FindDataFromBuffer**(**unsigned char **\***buffer**,** unsigned char buffer\_length**,** unsigned char data\_length**,** unsigned char **\*** read\_data**,** unsigned char send\_serial**)** **{**  unsigned int i**=**0**;**    **for** **(**i**;** i**<**buffer\_length**-**data\_length**;** i**++)** **{**  **if** **(**  buffer**[**0**+**i**]** **==** 3 **&&**  buffer**[**1**+**i**]** **==** 2 **&&**  buffer**[**5**+**i**]** **==** 3 **&&**  buffer**[**6**+**i**]** **==** 2  **)**  **{**  read\_data**[**0**]** **=** buffer**[**2**+**i**];**  read\_data**[**1**]** **=** buffer**[**3**+**i**];**  read\_data**[**2**]** **=** buffer**[**4**+**i**];**    **if** **(**send\_serial**)** **{**  transmit\_rf**(**buffer**[**2**+**i**]);**  transmit\_rf**(**buffer**[**3**+**i**]);**  transmit\_rf**(**buffer**[**4**+**i**]);**  **}**    **return** **(**  **((**buffer**[**2**+**i**]-**48**)** **\*** 100**)** **+**  **((**buffer**[**3**+**i**]-**48**)** **\*** 10**)** **+**  **((**buffer**[**4**+**i**]-**48**)** **\*** 1**)**  **);**  **}**    **}**    **return** 0**;**  **}** |
| Header |
| #ifndef LIBCOMPASS\_H  #define LIBCOMPASS\_H  unsigned char ReadCompass**(**unsigned char send\_serial**,** unsigned char **\*** read\_data**);**  unsigned int FindDataFromBuffer**(**unsigned char **\***buffer**,** unsigned char buffer\_length**,** unsigned char data\_length**,** unsigned char **\*** read\_data**,** unsigned char send\_serial**);**  #endif |

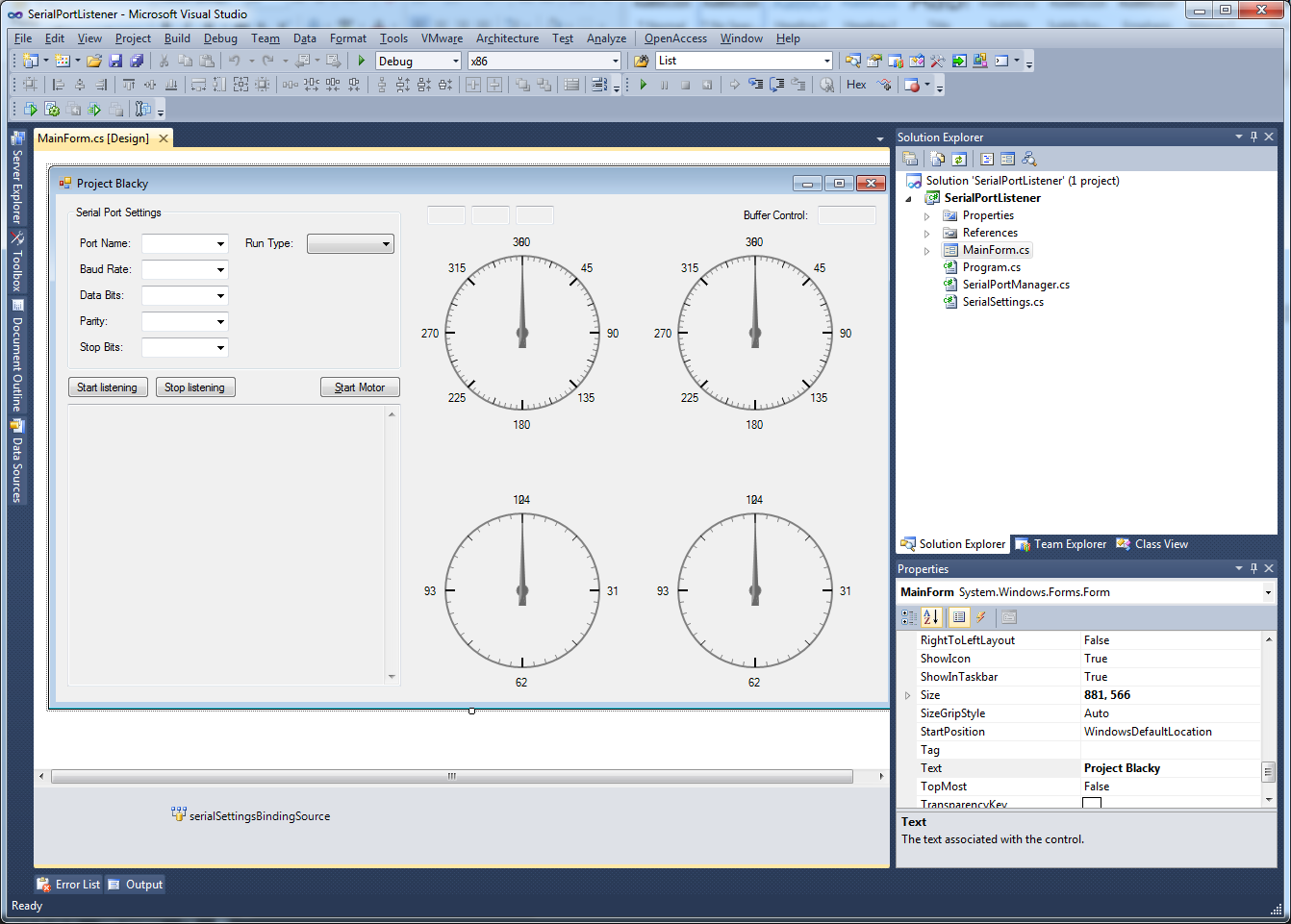
|  |
| --- |
| **NAVIGATION CONTROL AND MOTOR CONTROL**  libMotor.c – Contains code for Motor and Navigation Control. Some of the parameters were continuously edited throughout the process that in the final phase there were parameters that became useless but were not removed. |
| #include "libMotor.h"  void SetSpeed**(**unsigned char speed**)** **{**  TRISA **=** 0x00**;**  LATA **=** speed**;**  **}**  void TurnLeft**()** **{**  Turn **(**200**,** speed\_veryslow**,** 0**);**  **}**  void TurnRight**()** **{**  Turn **(**200**,** speed\_veryslow**,** 1**);**  **}**  void Turn**(**unsigned char delay**,** unsigned char speed**,** unsigned char direction**)** **{**  TRISB **=** 0x00**;**  TRISA **=** 0x00**;**    LATA **=** speed**;**  **if** **(**direction**)** **{**  LATB**.**B1 **=** 1**;**  LATB**.**B2 **=** 0**;**    LATB**.**B3 **=** 1**;**  LATB**.**B4 **=** 0**;**  **}** **else** **{**    LATB**.**B1 **=** 0**;**  LATB**.**B2 **=** 1**;**    LATB**.**B3 **=** 0**;**  LATB**.**B4 **=** 1**;**  **}**  **}**  void MoveForward **(**unsigned char speed**)** **{**  TRISB **=** 0x00**;**  TRISA **=** 0x00**;**    LATA **=** speed**;**    LATB**.**B1 **=** 1**;**  LATB**.**B2 **=** 0**;**    LATB**.**B3 **=** 0**;**  LATB**.**B4 **=** 1**;**  **}**  void MoveBackward **(**unsigned char speed**)** **{**  TRISB **=** 0x00**;**  TRISA **=** 0x00**;**    LATA **=** speed**;**    LATB**.**B1 **=** 0**;**  LATB**.**B2 **=** 1**;**    LATB**.**B3 **=** 1**;**  LATB**.**B4 **=** 0**;**  **}** |
| Header |
| #ifndef LIBMOTOR\_H  #define LIBMOTOR\_H  void SetSpeed**(**unsigned char speed**);**  void TurnLeft**();**  void TurnRight**();**  void Turn**(**unsigned char delay**,** unsigned char speed**,** unsigned char direction**);**  void MoveForward **(**unsigned char speed**);**  void MoveBackward **(**unsigned char speed**);**  // these variables are at Blacky.h  extern unsigned char speed\_veryfast**;**  extern unsigned char speed\_veryslow**;**  #endif |

|  |
| --- |
| **WIRELESS COMMUNICATIONS**  libRF.c – Handling of Wireless RF Transceiver RX/TX functions. |
| unsigned int transmit\_rf**(**unsigned char input**)** **{**  **if** **(**Soft\_UART\_Init**(&**PORTD**,** 6**,** 7**,** 9600**,** 0**)** **==** 0**)** **{**  Soft\_UART\_Write**(**input**);**  **return** 0**;**  **}** **else** **{**  **return** 1**;**  **}**  **}**  unsigned char read\_rf**(**unsigned char **\***error**)** **{**  **if** **(**Soft\_UART\_Init**(&**PORTD**,** 6**,** 7**,** 9600**,** 0**)** **==** 0**)** **{**  **return** Soft\_UART\_Read**(**error**);**  **}** **else** **{**  **return** 0**;**  **}**  **}** |
| Header |
| #ifndef LIBRF\_H  #define LIBRF\_H  unsigned int transmit\_rf**(**unsigned char input**);**  unsigned char read\_rf**(**unsigned char **\***error**);**  #endif |

|  |
| --- |
| **MAIN MODULE**  Blacky.c – contains code for Obstacle Avoidance, heading resolution, and wireless communications. |
| #include "PIC18F4550lib.h"  #include "libCompass.h"  #include "corePing.h"  #include "Blacky.h"  #include "libRF.h"  #include "libMotor.h"  void main**()** **{**  ADCON1 **|=** 0x0F**;**  CMCON **|=** 7**;**  main\_init**();**  TRISD **=** 0x2A**;**  TRISB **=** 0x00**;**  LATB **=** 0x00**;**  LATD **=** 0**;**  UART1\_Init**(**9600**);**    Delay\_ms**(**100**);**    // initialize defaults.  MoveForward **(**0**);**  test **=** 0**;**  cycle\_compass **=** 0**;**  start **=** 0**;**    **while(**1**){**  err **=** ReadPing**(&**left**,** **&**front**,** **&**right**);**    transmit\_rf**(**1**);**  transmit\_rf**(**left**);**  transmit\_rf**(**2**);**  transmit\_rf**(**right**);**  transmit\_rf**(**3**);**  transmit\_rf**(**front**);**  transmit\_rf**(**4**);**    **if** **(**enable\_compass **>** 0**)** **{**  **if** **(**enable\_compass **==** 1**)** **{**  current\_direction **=** ReadCompass**(**1**,** compass\_current**);**  **}** **else** **if** **(**enable\_compass **==** 2**)** **{**  **if** **(**cycle\_compass **>** COMPASS\_SPARCE**)** **{**  current\_direction **=** ReadCompass**(**1**,** compass\_current**);**  cycle\_compass **=** 0**;**  **}**  cycle\_compass**++;**  **}**  **}** **else** **{**  transmit\_rf**(**compass\_initial**[**0**]);**  transmit\_rf**(**compass\_initial**[**1**]);**  transmit\_rf**(**compass\_initial**[**2**]);**  **}**    transmit\_rf**(**5**);**  transmit\_rf**(**start**);**  transmit\_rf**(**6**);**    **if** **(**start **>** 0**)** **{**  transmit\_rf**(**compass\_initial**[**0**]);**  transmit\_rf**(**compass\_initial**[**1**]);**  transmit\_rf**(**compass\_initial**[**2**]);**  **}** **else** **{**  transmit\_rf**(**compass\_current**[**0**]);**  transmit\_rf**(**compass\_current**[**1**]);**  transmit\_rf**(**compass\_current**[**2**]);**  **}**    transmit\_rf**(**current\_direction**);**  transmit\_rf**(**initial\_direction**);**    **if** **(**start **==** 0**)** **{**  configuration **=** read\_rf**(&**temp**);**  **if** **(**configuration **>** 0**)** **{**  start **=** 1**;**  initial\_direction **=** ReadCompass**(**0**,** compass\_initial**);**  **}**  **if** **(**configuration **==** 1**)** **{**  speed\_veryfast **=** 0xFF**;** // should be max speed  speed\_veryslow **=** 3**;**  enable\_compass **=** 2**;** // sparce compass (every COMPASS\_SPARCE cycle);  **}** **else** **if** **(**configuration **==** 2**)** **{**  speed\_veryfast **=** 0xFF**;** // should be max speed  speed\_veryslow **=** 3**;**  enable\_compass **=** 1**;** // compass reading after each cycle  **}** **else** **if** **(**configuration **==** 3**)** **{**  speed\_veryfast **=** 5**;**  speed\_veryslow **=** 2**;**  enable\_compass **=** 2**;** // sparce compass (every COMPASS\_SPARCE cycle);  **}** **else** **if** **(**configuration **==** 4**)** **{**  speed\_veryfast **=** 5**;**  speed\_veryslow **=** 2**;**  enable\_compass **=** 1**;** // compass reading after each cycle  **}** **else** **if** **(**configuration **==** 5**)** **{**  speed\_veryfast **=** 0xFF**;**  speed\_veryslow **=** 3**;** // should be max speed  enable\_compass **=** 0**;** // no compass after each cycle  **}** **else** **if** **(**configuration **==** 6**)** **{**  speed\_veryfast **=** 5**;**  speed\_veryslow **=** 2**;**  enable\_compass **=** 0**;** // no compass after each cycle  **}** **else** **{**  speed\_veryfast **=** 0**;**  speed\_veryslow **=** 0**;**  enable\_compass **=** 0**;** // no compass after each cycle  **}**  **}**    **if** **(**start **>** 0**)** **{**  **if** **(**front **<** MIN\_SAFE\_DISTANCE **&&** right **>=** MIN\_SAFE\_DISTANCE **&&** left **>=** MIN\_SAFE\_DISTANCE**)** **{**  **if** **(**right **>** left**)** **{**  **if(**cycle\_compass **!=** COMPASS\_SPARCE**)** **{**  TurnRight**();**  **}** **else** **{**  MoveBackward**(**speed\_veryfast**);**  Delay\_ms**(**400**);**  TurnRight**();**  Delay\_ms**(**100**);**  **}**  **}** **else** **{**  **if(**cycle\_compass **!=** COMPASS\_SPARCE**)** **{**  TurnLeft**();**  **}** **else** **{**  MoveBackward**(**speed\_veryfast**);**  Delay\_ms**(**400**);**  TurnLeft**();**  Delay\_ms**(**100**);**  **}**  **}**  **}** **else** **if** **(**right **<** MIN\_SAFE\_DISTANCE **&&** front **<** MIN\_SAFE\_DISTANCE **&&** left **>=** MIN\_SAFE\_DISTANCE**)** **{**  **if(**cycle\_compass **!=** COMPASS\_SPARCE**)** **{**  TurnRight**();**  **}** **else** **{**  MoveBackward**(**speed\_veryfast**);**  Delay\_ms**(**400**);**  TurnRight**();**  Delay\_ms**(**100**);**  **}**  **}** **else** **if** **(**left **<** MIN\_SAFE\_DISTANCE **&&** front **<** MIN\_SAFE\_DISTANCE **&&** right **>=** MIN\_SAFE\_DISTANCE**)** **{**  **if(**cycle\_compass **!=** COMPASS\_SPARCE**)** **{**  TurnLeft**();**  **}** **else** **{**  MoveBackward**(**speed\_veryfast**);**  Delay\_ms**(**400**);**  TurnLeft**();**  Delay\_ms**(**100**);**    **}**  **}** **else** **if** **(**left **<** MIN\_SAFE\_DISTANCE **&&** front **<** MIN\_SAFE\_DISTANCE **&&** right **<** MIN\_SAFE\_DISTANCE**)** **{**  MoveBackward**(**speed\_veryfast**);**  **}** **else** **if** **(**front **>** MIN\_SAFE\_DISTANCE **&&** left **>** MIN\_SAFE\_DISTANCE **&&** right **>** MIN\_SAFE\_DISTANCE**)** **{**  **if(**abs**(**initial\_direction **-** current\_direction**)** **>** 4**)** **{**  TurnRight**();**  **}** **else** **{**  MoveForward**(**speed\_veryfast**);**  **}**  **}** **else** **if** **(**front **>** MIN\_SAFE\_DISTANCE**)** **{**  MoveForward**(**speed\_veryfast**);**  **}** **else** **{**  MoveBackward**(**speed\_veryfast**);**  Delay\_ms**(**400**);**  **}**  **}**    **if** **(**test **==** 0**)** **{**  LATB**.**B0 **=** 0x01**;**  test **=** 1**;**  **}** **else** **{**  test **=** 0**;**  LATB**.**B0 **=** 0x00**;**  **}**    //Delay\_ms(1000);  **}**  **}** |
| Header |
| #ifndef BLACKY\_H  #define BLACKY\_H  #define MIN\_SAFE\_DISTANCE 20  #define MIN\_SLOW\_DISTANCE 35  #define MIN\_LONG\_DISTANCE 60  #define COMPASS\_SPARCE 6  sbit Echo\_L at PORTD**.**B1**;**  sbit Trigger\_L at LATD**.**B0**;**  sbit Echo\_F at PORTD**.**B3**;**  sbit Trigger\_F at LATD**.**B2**;**  sbit Echo\_R at PORTD**.**B5**;**  sbit Trigger\_R at LATD**.**B4**;**  unsigned int left**,** right**,** front**,** err**;**  unsigned int test**;**  unsigned char soft\_uart\_error**;**  unsigned char soft\_uart\_read\_error**;**  unsigned char initial\_direction**;**  unsigned char current\_direction**;**  unsigned char compass\_initial**[**3**];**  unsigned char compass\_current**[**3**];**  // Configuration  unsigned char speed\_veryfast**;**  unsigned char speed\_veryslow**;**  unsigned char configuration**;**  unsigned char start**;**  unsigned char temp**;**  unsigned char enable\_compass**;**  unsigned char cycle\_compass**;**  #endif |

**SOFTWARE MONITORING CLIENT**

The software was developed using Visual Studio. Source files are posted on a link in an appendix page near the end of the paper.



The user chooses the correct port name, other connection related data, and the run type.

**RESULTS AND DISCUSSION**

Project Blacky was tested during a contest among other autonomous vehicles. Blacky was able to cross the contest field successfully after avoiding obstacles and rechecking heading. However, because Blacky carried 2 9V Batteries and 6 AA Batteries, he was relatively heavy and was running very slow.

**RECOMMENDATION**

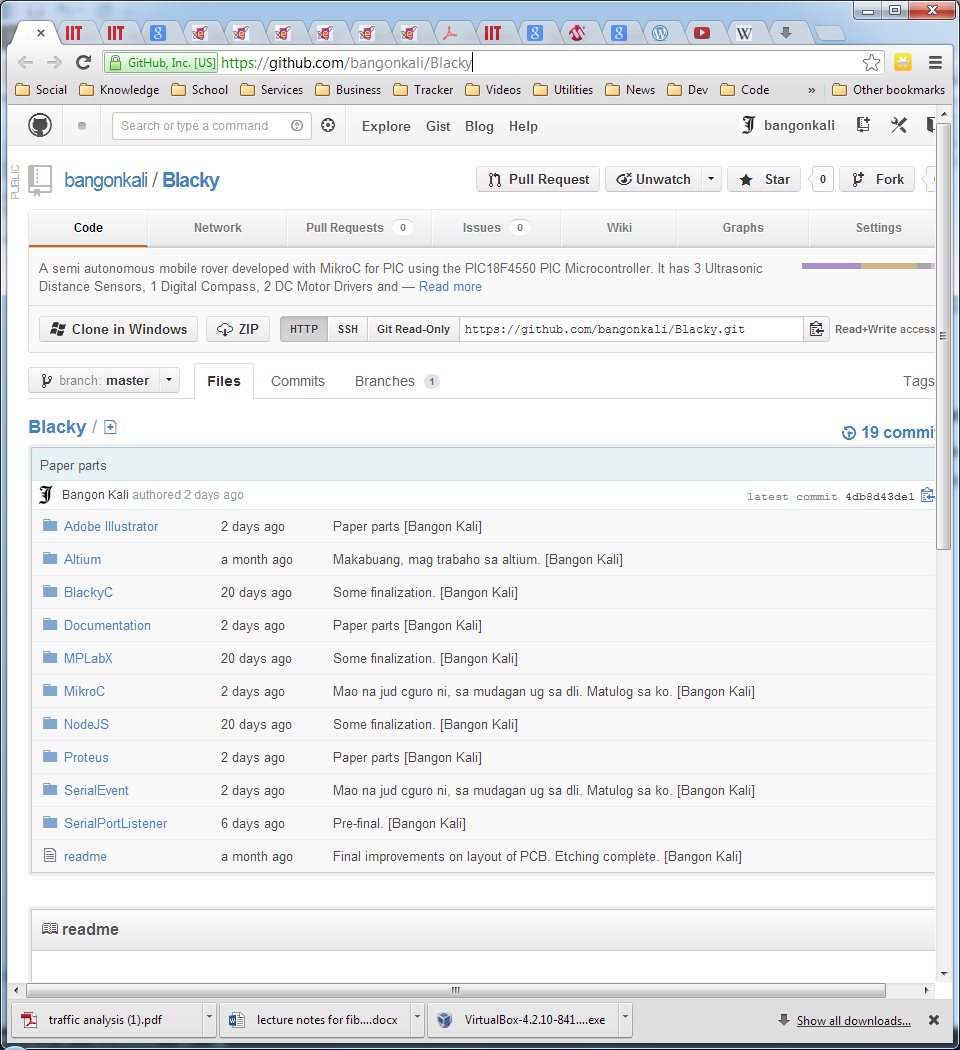
For future developers of a system comparable to Project Blacky, it is highly recommended to use improved algorithms in Obstacle Avoidance and Direction Heading Resolution. Also, the power requirement for Blacky was relatively heavy, a lighter and more stable power solution is also highly advisable.

**CONCLUSION**

Blacky was able to traverse the contest area successfully albeit very slow. It is therefore important that weight and power requirements be carefully considered during development of a project similar to Project Blacky. However, the success of Project Blacky is an example that simple autonomous robotic systems can be easily developed using off the shelf components and that all that matters is how advance and efficient the algorithm is.

**APPENDIX A**

Blacky will be an open source project. All code and related resources will be publicly accessible online through the internet via <https://github.com/bangonkali/Blacky>.



**APPENDIX B**

|  |  |
| --- | --- |
| US-100 Ultrasonic Sonar | <http://www.e-gizmo.com/KIT/sonar.htm> |
| Digital Compass  DIY Robot Navigation Aid | <http://www.e-gizmo.com/KIT/compass.htm> |
| Model 863 RF Wireless  Data Tranceiver | <http://www.e-gizmo.com/KIT/863.html> |
| 2 Channel DC Motor Driver  with Speed Control | <http://www.e-gizmo.com/KIT/2%20channel%20DC%20motor%20driver%20with%20speed%20control.html> |
| PBOT 2r0  Entry Level Mobile Robot Kit | <http://www.e-gizmo.com/KIT/P-BOT.htm> |
| PIC18F4550 Data Sheet | <http://ww1.microchip.com/downloads/en/devicedoc/39632e.pdf> |