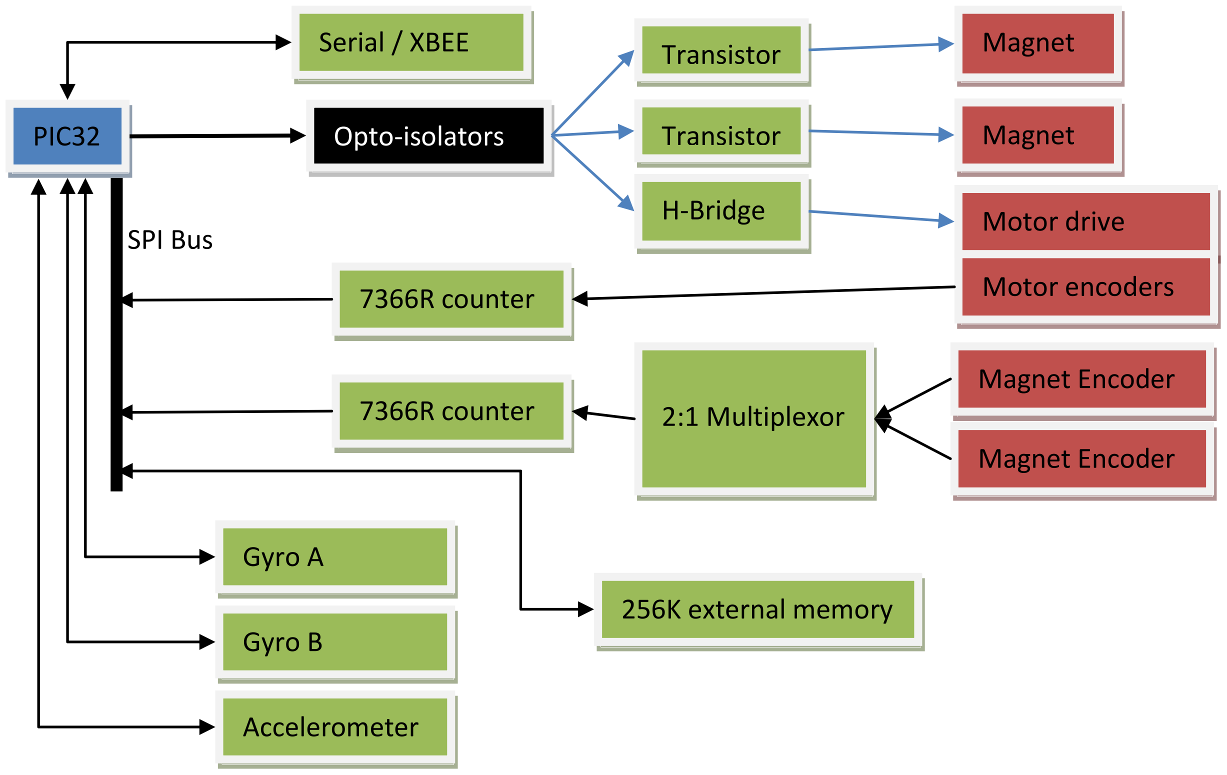
Monkeybot circuit and software notes

Part 1- hardware / circuitry notes

Overview

The Monkeybot is a two-link robot with a motor between the two links and an electromagnet on the end of each link. The Monkeybot is controlled by a Microchip PIC32MX460F512L which is connected to various circuitry to enable it to control the robot and collect input from encoders, accelerometers, gyros, and communicate with a wireless serial connection.

The following is a block diagram of the system:



Circuitry overview

The Monkeybot circuitry is composed of two boards. The main board, located on the motor link, is a manufactured PCB that is housed in a plastic box. The main board contains the main microcontroller, a PIC32MX460F512L, several support integrated circuits, and connections to sensors and other devices. The second board is a small PCB which is located on the shaft link and contains connections to peripheral devices meant to be located on the shaft link. The small PCB has connections for one battery, one magnet, one encoder, one gyro, and one accelerometer. The two PCBs are meant to be connected by a single large cable. The large cable splits into two ribbon cables, which are plugged into 10-pin connectors on the smaller board. The cable with the red wire should be plugged into the same side of the board as the magnet connector.

Most peripheral devices are plugged into the main board's box, and inside the box the panel connecter is connected to another plug that plugs into the PCB itself. This way, all the PCB plugs can be unplugged and the PCB can be removed. The one item that does not fit into this pattern is the on/off switch, which is directly connected to the PCB and must be removed from the box when the PCB is removed.

See Appendix A for a list of all parts used in the circuitry.

Batteries

There are two battery connector barrel plugs (one on the box and one on the lower PCB) which are currently connected to an 11.2V battery each. The center pin on the barrel plug is positive on both plugs. In the box the batteries are connected together in serial and run through the power switch. This voltage is used to power the motors and magnets, so the batteries could be changed as long as they were within the motor and magnet rating.

The 9V battery plugs directly into the PIC board (also through the main switch).

Magnets

There are two magnet barrel plugs, one each on the box and lower PCB. The magnets draw 250mA each, and are driven by transistors that can handle 8A (ST13007).

Encoders

There are two 10-pin headers on the main PCB for the magnet encoders. One is connected to the header on the outside of the box for the motor link encoder. The other goes through the large cable to the lower board, which also has a 10-pin header. The plugs on the encoders will only plug in one way on the small PCB. On the main PCB, the headers are at right angles because the PIC32 board covers them otherwise. As currently crimped, the 10 pin headers should be plugged into the board so that the nub on the plug is to the top.

Gyros

There are two gyro ports on the PCB, each with five pins that are in the same order as on the gyros themselves. Both the PCB and the lower board have black marker on one side of the plug. The line that carries power (3.3v) to the gyro is the pin furthest from the black marker. Thus, on the lower board the gyro should be plugged in the same direction, with the power line furthest from the black line. In the box the 5-pin plug coming from the big cable and marked "G" should be plugged with its black wire to the side with the marker.

Accelerometer(s)

There is one accelerometer port. The “A” plug coming from the inter-PCB wire should be plugged into the accelerometer port with the black wire to the black marker; on the lower board the connections are (from the black bar) ground, y, x, 3.3 V, ground. The accelerometer on the lower board should be plugged in so that the pins line up this way. The accelerometer and gyro ports on the main PCB are connected to power, ground, and PIC analog input pins.

For a second accelerometer, a plug has to be added that wires into the extra (unused) analog pins on the main PCB, because at present there is only one dedicated accelerometer port.

Motor

The motor connector on the main PCB is a DE9 (D-Sub) serial connector. Because of space constraints a normal D-Sub plug will not fit on the PCB because neighboring plugs. This is why the board currently has only the center part of the plug mounted on it.

The plug contains nine wires, 4 are for the main motor drive in two pairs of two, 4 other wires are used for the encoder (+5, ground, A, and B), and one pin is unused. The D-Sub plug is connected to a plug on the side of the box, where the motor itself can be plugged in. Notes on the power wiring are below in another section.

The motor uses around 2.3A maximum, according to the datasheet. The H-bridge can supply 4A using both channels in parallel, which is the way it is connected.

Magnets

The two magnets are controlled by NPN transistors which are controlled by the PIC’s digital output pins; these pins can be configured in software to function in on/off mode or in PWM mode. The plugs on the main board can currently only be plugged in one way. The wires for the plugs lead to the side of the box for one magnet and the lower board for the other.

7366 chips

There are two 7366R counters for counting encoder pulses- one for the motor and one for the magnet encoders. The magnet encoders go through a multiplexor so that only one is connected to the 7366R at a given time. When switching the magnets, it may be desirable to load a new value into the 7366R counter register, since because of the geometry a different starting angle value may be appropriate.

The 7366Rs are on the PIC’s SPI1 bus, but have unique chip select lines (active low). The support code is already written, with the use described in a later section of this document.

RAM

The RAM is on the same SPI line as the 7366 chips, again with a unique chip select line. Using the RAM will require looking at the datasheet and writing new support code, since it hasn’t been done yet.

XBEE / serial out

There is a 6 pin header on the PCB which is meant to mate directly to the 6 pin output of the USB-RS232 cable. If this cable is used, the jumper next to the header **must** be removed. If instead you want to plug the XBEE directly into the PIC, the jumper should be inserted for 5v power, and a small cable can be used to connect the XBEE to the PCB. The side with ground is marked with black marker. Note that the cable used with the XBEE must “crossover”, that is the TX and RX lines should be switched on one side. The cable currently with the box is already like this.

Notes on the different power planes

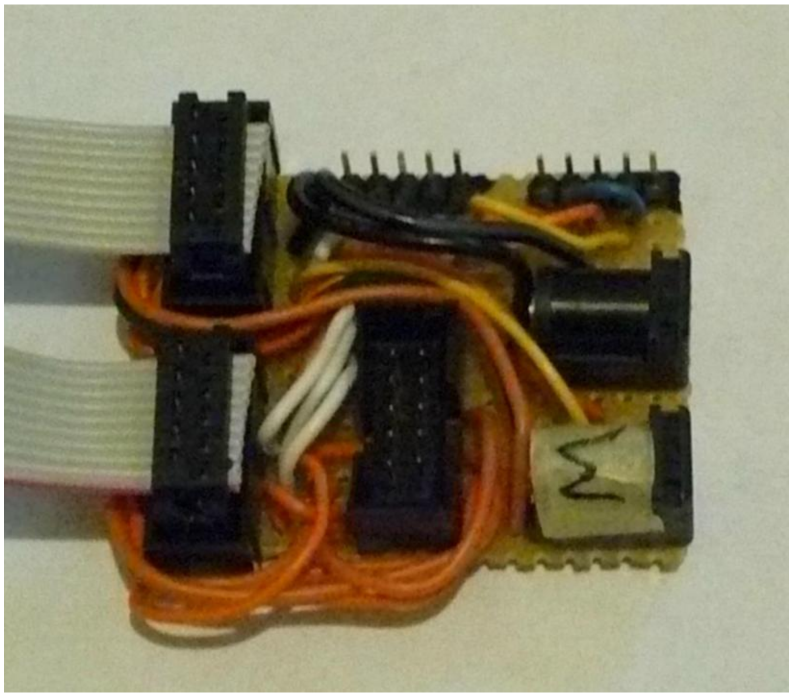
The main PCB has two distinct power planes that are not connected in any way- the main electronics plane and a plane for the motor and magnets. The main electronics plane is connected to the PIC and all its peripheral devices, like the gyro, accelerometers, and encoders (see schematic). This plane has a 5V and a 3.3V line, and takes its power from the 9V battery.

The other plane is for the 24V devices- the magnets and the motor. The devices running off this plane are the motor, the H-bridge, and the magnets (and magnet transistors). This plane gets its power from the two rechargeable batteries.

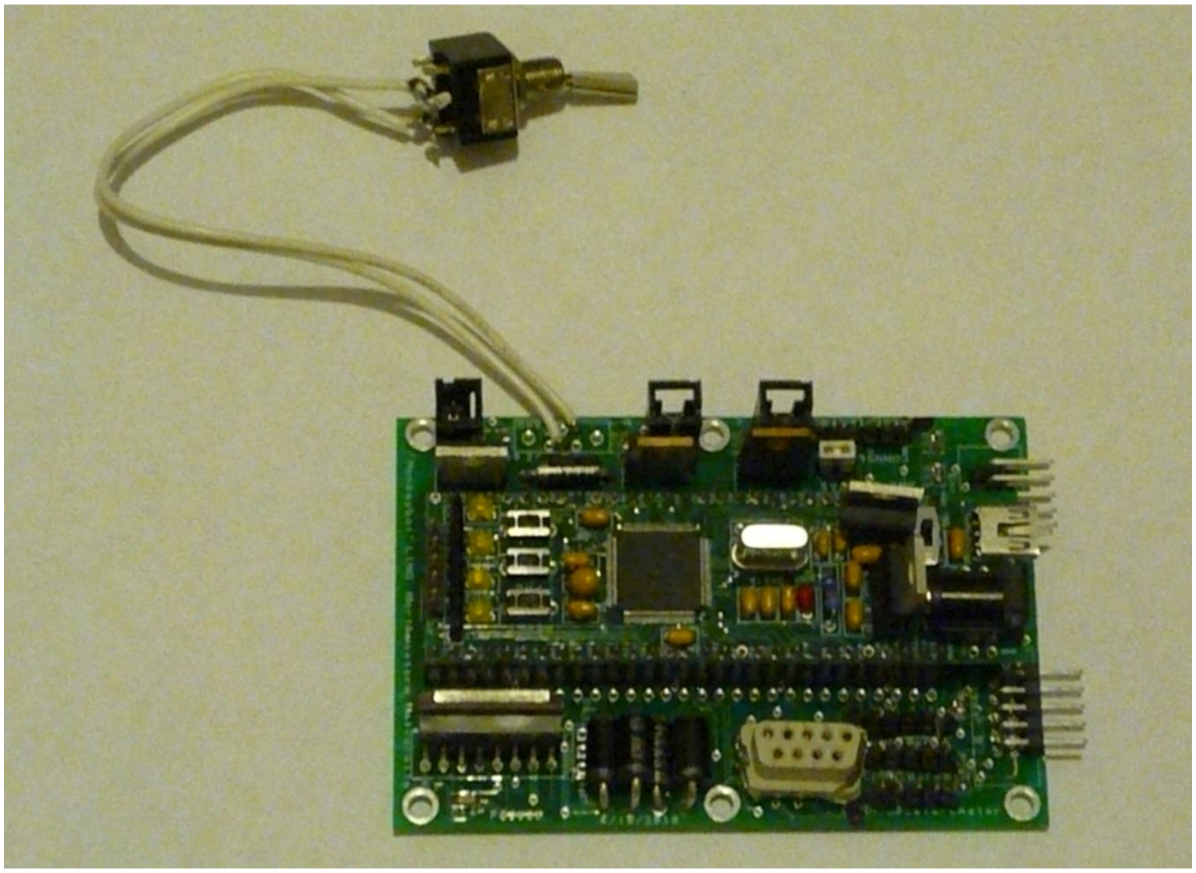
The two planes are not directly connected (not even the ground wire). They connect using on-board optical isolators. There are 5 lines that are optically isolated- the 3 h-bridge lines (enable, A, B) and the two magnet control lines. There are two opto-isolator chips: one with 4 opto-isolators and one with 1.

Note that the motor plug has lines running to both planes- the encoder is on the main circuitry plane and the motor itself is on the motor plane.

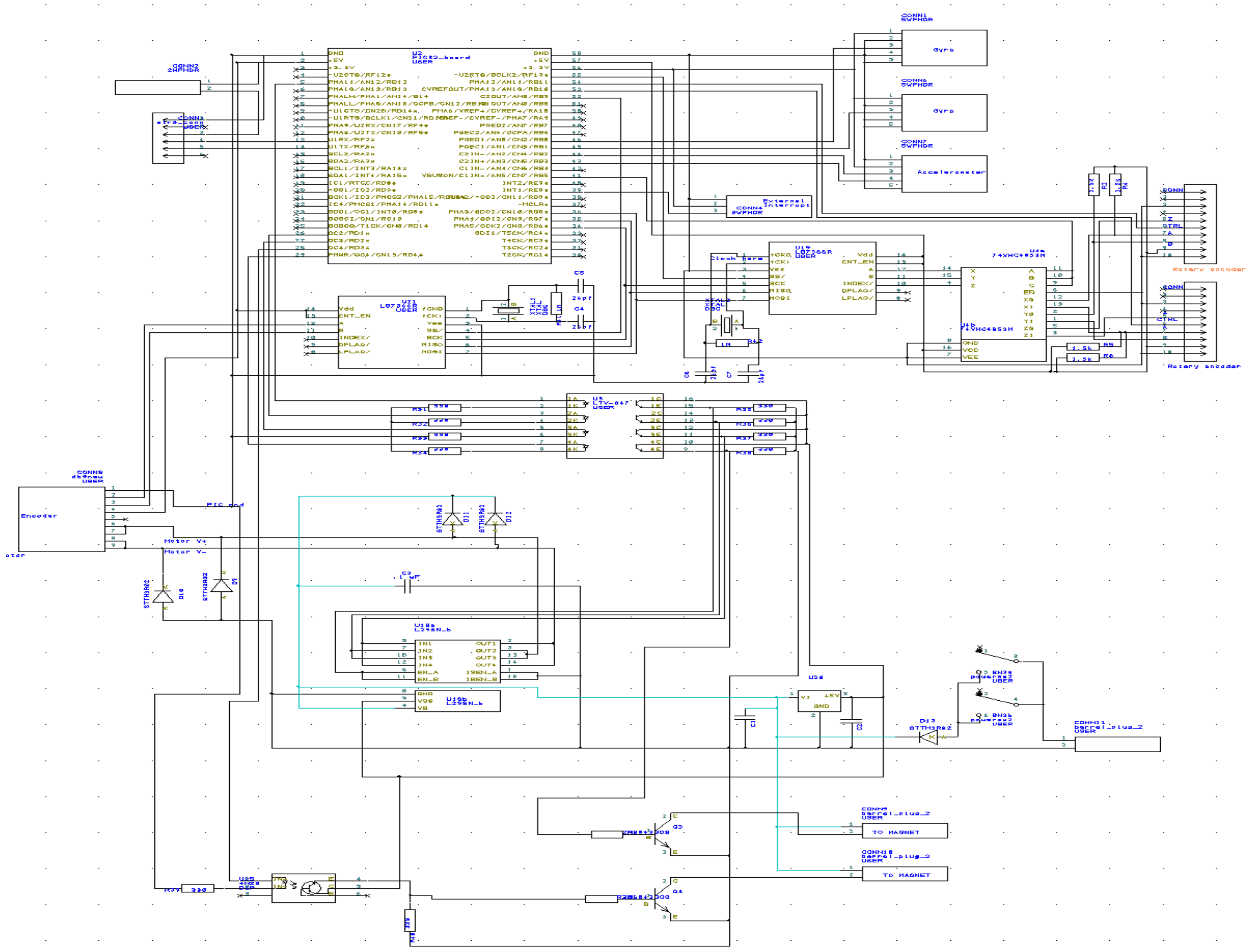
Small protoboard:



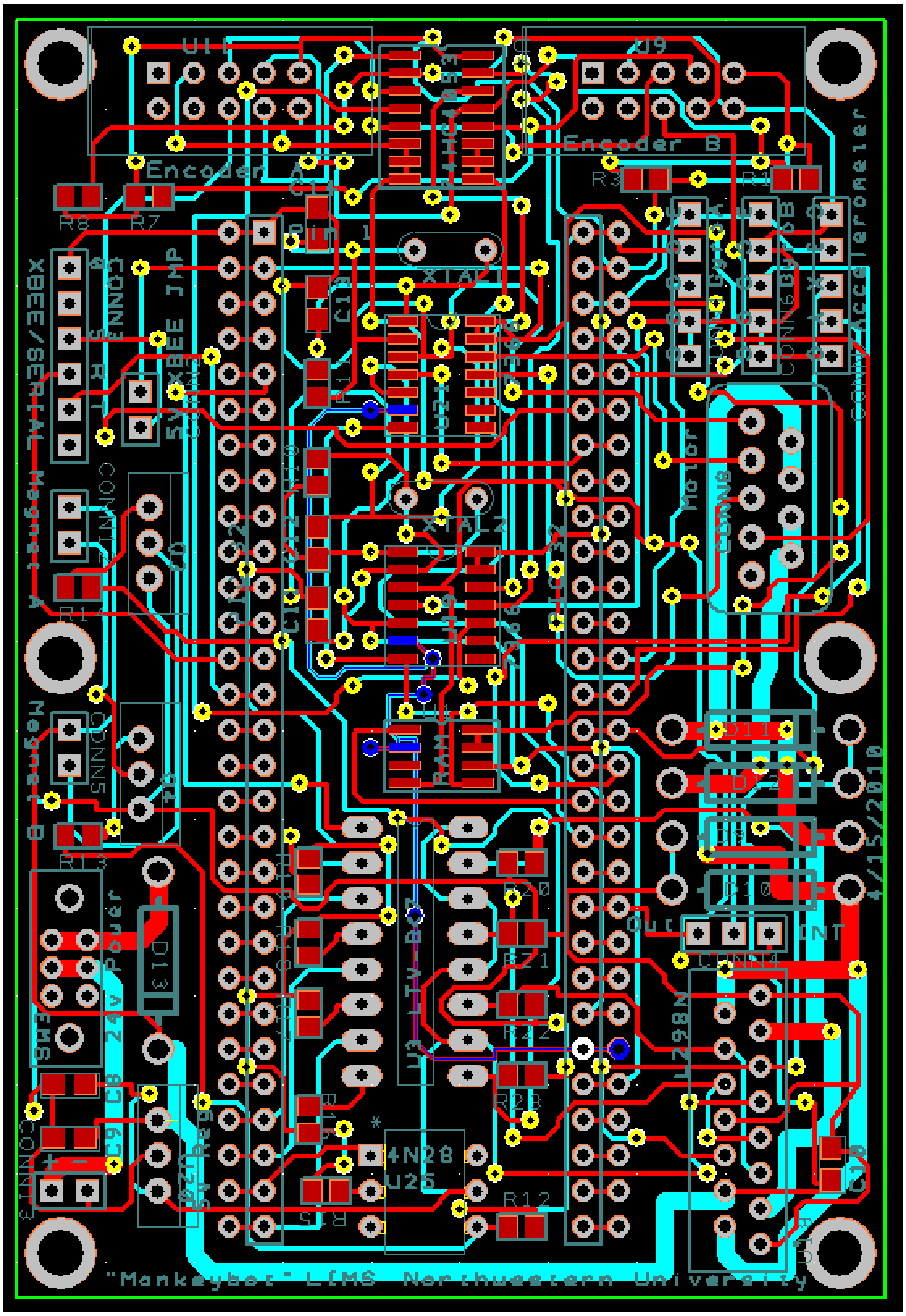
Main PCB:



Schematic:

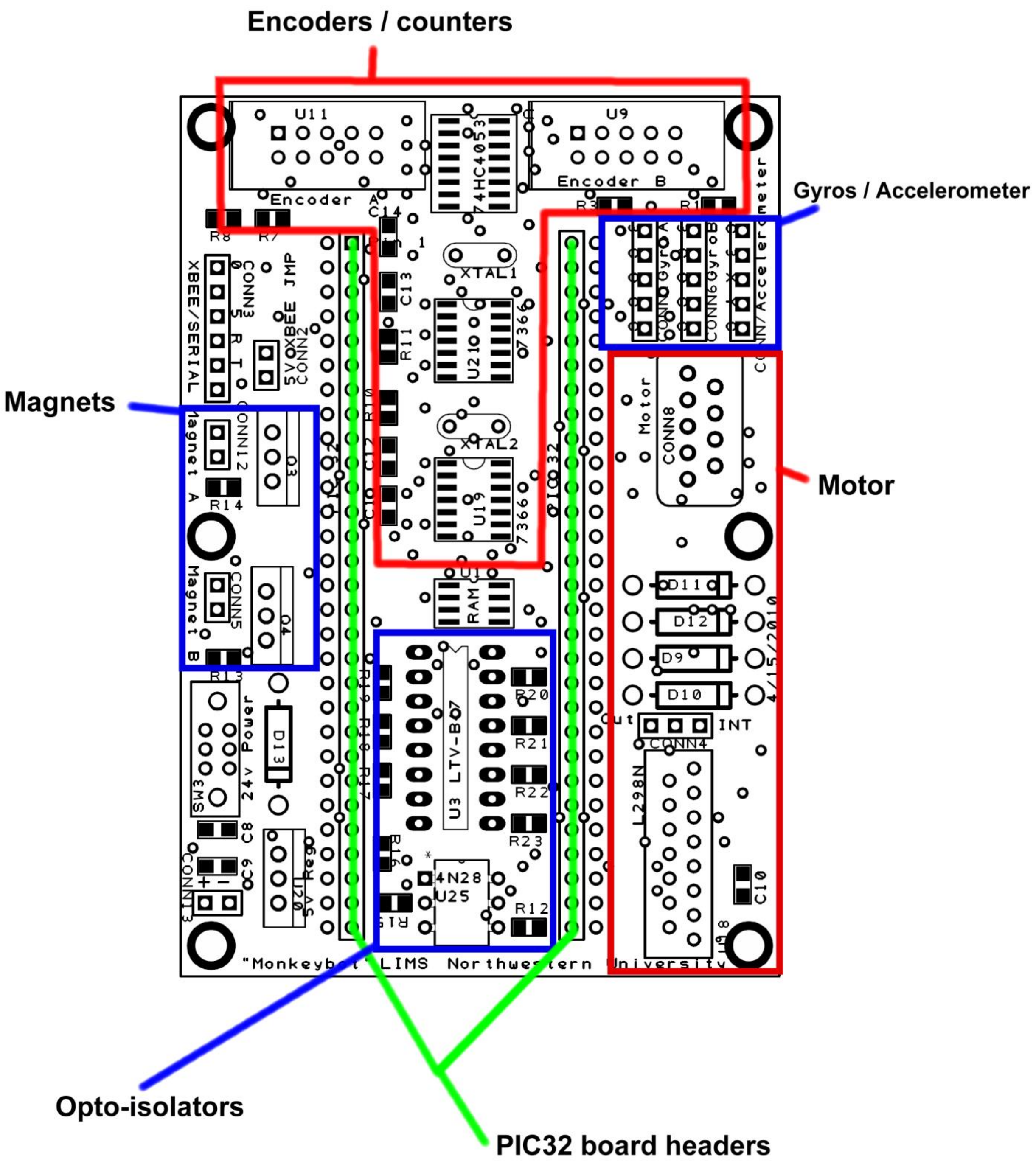

Gyro
Gyro
Accelerometer
External Interrupt
Rotary encoder
Rotary encoder
Encoder
Motor V+
Motor V-
PIC gnd
TO MAGNET
TO MAGNET

PCB:



U11
U9
Encoder A
Encoder B
R8
R7
R3
R1
XTAL
XTAL2
XBEE/SERIAL
XBEE JMP
5V
Magnet A
Magnet B
Motor
Accelerometer
Power
24v
D13
4N28
U25
L298N
D11
D12
D10
R12
R20
R21
R22
R23
CONN4
"Monkeybot" LIMS Northeastern University
4/15/2010

PCB Map:



Encoders / counters
Gyros / Accelerometer
Magnets
Motor
Opto-isolators
PIC32 board headers
U11
U9
74HC4053
Encoder A
Encoder B
XTAL1
XTAL2
U21
U19
RAM
U3 LTV-847
4N28
U25
XBEE/SERIAL
Magnet A
Magnet B
24v Power
L298N
CONN8
CONN4
"Monkeybot" LIMS Northwestern University
4/15/2010

Part 2- Software notes

This section contains notes on how to use the various peripheral devices in software, as currently supported.

Pin list

The following is a list of the pins connected to various devices on the PIC.

UART1 (RF8 TX, RF2 RX)- XBEE / serial
OC2/OC3 (RD1 / RD2)- Hbridge1 and Hbridge2 control
OC4 (RD3) – Magnet A
OC5 (RD4) – Magnet B
AN0 (RB0) – Gyro A
AN1 (RB1) – Gyro B
AN2 (RB2) – Accelerometer X
AN3 (RB3) – Accelerometer Y
SPI2 (RG8 SDO2, RG7 SDI2, RG6 SCK2) – SPI devices (two 7366s and RAM)
RB12 – H-bridge enable
RF12 – Slave select for magnet 7366R
RB9 – Slave select for motor 7366R
RB11 – Control bit for magnet encoder A
RB10 – Control bit for magnet encoder B
RB5 – MUX control bit

Motor

The motor is connected to the PIC via three lines- the H-bridge enable and the two H-bridge input pins. The two input pins go to PWM-enabled ports. There is a function called `configure_pins()` that does the necessary pin setup. As currently programmed, one input pin functions as pin direction and the other is used for PWM. As with all output pins, there are `#define`s already made that allow easy pin access. For instance:

```
HBRIDGE_EN = 0;
```

Is all that is needed to disable the h-bridge. The direction pin can be similarly changed with `HBRIDGE_DIR`. To set the PWM rate, the function `SetDCOC1PWM(duty)` can be used. The max duty is currently 0xFFFF (65,535).

None of these have to be used though, because there are support functions that can be used instead. The function `stop()` stops the motors immediately. The function `set_speed(speed)` can be passed an integer for “speed” that is from -65,535 (full reverse) to 65,535 (full forward).

Magnets

The magnets are currently very easy to use, since they use digital outputs. The `configure_pins()` function that was mentioned earlier sets up the pins as needed. Then the outputs can be used easily with the defines `MAGNET_A` and `MAGNET_B`. For instance,

```
MAGNET_A = 1;
```

turns magnet A on. At this point, the pin is high, and will remain high until `MAGNET_A = 0` is executed. The magnet outputs are on PWM pins so in the future the program could be modified to setup and use the magnets with PWM instead of on-off control.

Accelerometers / Gyros

The accelerometers and gyros use analog output to send data to the PIC. Each gyro has one analog out, and the encoder has two analog outs. There are simple support functions which can be called to return the current reading from one of these as an int. The reading is out of 1024. This is a list of the functions:

```
int read_gyroA();
int read_gyroB();
int read_acc1x();
int read_acc1y();
```

XBEE / serial

The serial out used for the XBEE is on UART1. Thus, all that needs to be done to output serial data to the XBEE is to use the function `PutSUART1()`. For instance,

```
sprintf(string,"Read from CNTR: %x %x\r\n", variable1, variable2);
putsUART1(string);
```

Will write the given string to “string”, then put it out on UART1.

Encoders / 7366R counters

There are three encoders in the system- two magnet encoders and the motor encoder. There are also two counters- one for the motor and one for the magnets. When the magnets are switched, the MUX for the motor encoders should also be switched (using `MUX_CTRL = x`) and the software should clear or change the counter.

At startup, the function `setup_counters()` should be called to setup the counters. This function initializes the counters and sets the DTR register, which in modulo-n mode contains the encoder high count rollover point. The numbers are currently correct for the current encoders. If they ever need to be changed, the number in the setup function should be changed.

During normal operation, the current position of the motor can be obtained using the functions `read_magnet_encoder()` and `read_motor_encoder()`, which each return integers. The support functions work by setting a global variable called `current_encoder`, which causes calls to the 7366 library functions to know which chip enable line to use. If the counter register ever needs to be changed (such as when the magnet encoders switch and thus the encoder is starting at a new angle), this is how it is done:

```
writebuf[1] = 0x23;
writebuf[0] = 0x01;
write_7366(CNTR, writebuf); //Sets up motor CNTR to contain 0x0123
```

Note that writebuf is passed as a pointer, the library functions assume the buffer size is the same as the number of bytes the counter is set to operate at (set in 7366.h). In the default mode, including the code shown above, two byte mode is used. Note also that the **most** significant byte is in writebuf[0], and the **least** significant byte is in writebuf[n], where n is the number of bytes the counter is operating in. If the CNTR register only needs to be cleared, the following command can be used instead:

```
clear_reg_7366(CNTR);       //Sets CNTR to 0
```

RAM chip

The PCB has an extra onboard RAM chip, the ON Semiconductor N25S830HAS22I. The RAM chip is 256Kbit (32Kbyte). It is on the SPI1 bus with the 7366R counters, but with its own chip enable (active low) on pin B4. This is not currently implemented in the program and support has to be added. The datasheet is available at http://www.onsemi.com/pub_link/Collateral/N25S830HA-D.PDF

See Also

Google code page http://code.google.com/p/hoppingrobot/

7366R wiki page
http://hades.mech.northwestern.edu/index.php/Using_the_LS7366R_SPI_Quadrature_Counter

ME333- Monkeybot wiki page http://hades.mech.northwestern.edu/index.php/Monkeybot

Appendix A- parts list

Note that “quantity” refers to the exact number needed for the circuit.

ON BOARD PARTS

Qty	Part	Digikey P/N
1	MUX : M74HCT4053RM13TR	497-7398-1-ND
2	Crystal : HC49US-40.000MABJ-UB	300-8520-ND
1	Opto-isolator : LTV-847	160-1370-5-ND
5	Diode : STTH3R02RL	497-5257-1-ND
1	H-bridge : L298N	497-1395-5-ND
1	5V reg : LM2940	LM2940CT-5.0-ND
2	Transistor : 2N6043G	2N6043GOS-ND
1	opto-isolator : 4N27	4N27-ND
12	330 ohm resistors	RMCF1/10330FRCT-ND
2	1 Mohm resistors	RMCF1/101MJRDKR-ND
4	1.5 kohm resistors	RMCF1/101.5KFRDKR-ND
1	22 uf Capacitor	PCC2401CT-ND
1	.47 uf capacitor	399-5729-6-ND
1	.1 uf capacitor	478-3351-6-ND
4	26 pf capacitor	PCC270CGCT-ND
1	256k RAM : N25S830HAS22I	766-1043-ND
2	Rectangular headers for PIC32	S7062-ND
Non-Digikey:		
2	Decoder/counter: 7366R	US Digital p/n: LFLS7366R-S

PLUGS

Board mounted sockets:

Qty	Digikey P/n
3	A33923-ND

Board-panel plugs

Qty	Digikey p/n
9	A34181-ND
3	A34187-ND

Panel-mount sockets

Qty	Digikey p/n
2	CP-011B-ND
1	CP-1238-ND
1	HPP10H-ND
1	H11330-ND

Plugs for panel-mount sockets

Qty	Digikey p/n
4	CP3-1001-ND
1	CP-1380-ND
1	H10050-ND

Appendix B: Additional code reference

Encoders

Here is a typical setup and use of the encoders, showing both used and converted to actual angles.

```
int main {
      //…
      //Set up encoder ships
      setup_counters();
      //…
      while (1) { //Main while loop
            int magnet_position = int read_magnet_encoder();      //0-0x3840
            float magnet_position_radians =
               (float)(2*PI)*(float)(magnet_position)/(float)(0x3840);
            //…
            int motor_position = read_motor_encoder(); //Returns 0 – 0x9858
            float motor_position_radians =
               (float)(2*PI)*(float)(motor_position)/(float)(0x9858);
            //Use numbers for control…
      }
}
```

Note that the magnet encoders have a high stop of 0x3840, meaning that each count is 4.36 x 10^{-4} radians. The motor high stop is 0x9858, corresponding to 1.61 x 10^{-4} radians / count.

Gyros / Accelerometers

The accelerometer is meant to output an analog voltage on each channel, x and y. The nominal voltage is Vss/2, or 1.65V, which corresponds to a reading of 512. At this reading, the accelerometer is experiencing zero acceleration. For other readings, the accelerometer should change 0.62Volts/g at 3.3V. For instance, to find the X tilt from the reading, the code would be:

```
int reading = read_acc1x;
int tilt = asin( (reading-512) / (.62*1024/3.3) );
```

The gyros function in a similar way, with a reading of 512 again meaning zero degrees/second of angular rotation. The sensitivity is then 3.3mV / (deg/sec). To find the number of degrees/second, the code could be:

```
int reading = read_gyroA();
int DegreesPerSecond = (reading – 512) * 1000 / 1024;
```

Appendix C- Future improvements

Listed here are some possible future improvements that could be made to the electrical system.

- It would be nice to have the bottom PCB be fabricated. The cost is probably not worth it to order it alone, but if any other PCB has to be made adding the lower PCB to the order would be a good idea
- Although it looks nicer to have the electronics in a box, it would probably be best to have no box. Assembling and disassembling the circuit takes a great deal of time with the box, and it also isn't very good for airflow
- If the main PCB were ever changed, one definite improvement would be to move the two 10-pin encoder connectors so that they aren't blocked by the PIC32
- Another PCB improvement would be to add a header for a second accelerometer
- A third PCB improvement would be to slightly move the main power plug. As it is it partially covers the mounting hole, just enough so that it makes getting the screw in difficult