ROBOTIC MICROPHONE STAND FOR POGO STUDIO

By

Kai Jiang

Weihong Yuan

Final Report for ECE 445, Senior Design, Fall 2012

TA: Lydia Majure

10 December 2012

Project No. 28

Abstract

Mark Rubel from Pogo Studio in downtown Champaign introduced this project to us. As required by Mark, the robotic microphone stand we designed and built has the moving range of 3 feet in length, 2 feet in width and 3.5 feet in height. It is remote controlled and has four memory locations that can store and retrieve exact microphone positions. Our project is fully functional and it has met all requirements. This report details the design procedure and the results of our project.

Contents

1. Introduction

In this report, we will first introduce the purpose and functions of our project, and then describe the design of each module as a whole. Detailed design verification and a cost analysis will then be followed. Finally, we will discuss the success we have accomplished, possible future development, and any uncertainties or ethical considerations.

1.1 Purpose

In a recording environment, a difference in microphone placement of a few inches makes all the sonic difference. Usually, the sound source (micing instruments or amplifiers) is remote from the listening environment (in our case, 10 to 15 meters away). In order to compare microphone's positions, one has to either run in and out of the recording area while losing audio memory or have an assistant adjusting the instruments' positions in a potentially high-volume environment. For either case, one will have trouble remembering previous instruments' locations. The motivation behind this project is to design and build a remote controlled microphone-positioning robot with positional memories that can provide an efficient, more convenient and safer solution for the problems elaborated above.

Our project is separated into two parts: the control board and the microphone stand. On the control board, we have six push buttons corresponding to six movement directions (forward, backward, left, right, up and down), and three switches, which have the functions of power, store and retrieve. The six pushbuttons are also used as memory locations when the system is switched into store or retrieve mode. On the microphone stand side, the microphone is installed on a machine that is designed by us and built by the ECE machine shop. The machine has three tracks for horizontal movements (X, Y), and one additional track for vertical adjustment (Z). The microphone is mounted on the Z track. Thus, the microphone can be moved within the three-dimensional range.

1.2 Project Functions and Features

- Power supplies are portable and no more than 12V, which makes the microphone stand safe to work with and easy to carry around.
- Remote controlled via RF modules. It can be operated at least 20 meters away and with obstacles in between.
- User-friendly. The control board we designed is straightforward and easy to operate.
- Three-dimensional movements. The microphone can be moved forward, backward, left, right, up and down.
- Moving range is 3 feet in length, 2 feet in width and 3.5 feet in height.
- Up to six positional memories, which is as required by Mark.
- Small retrieving error (less than 0.3cm) to ensure positioning accuracy.
- Low cost. The whole project is inexpensive. We used mechanical motor encoders instead of optical motor encoders or motors with encoders pre-installed, which saved us at least 100 dollars. Also, only one PIC micro-controller instead of three is used to control the three motors, which saved about 15 dollars.

1.3 Block Diagram

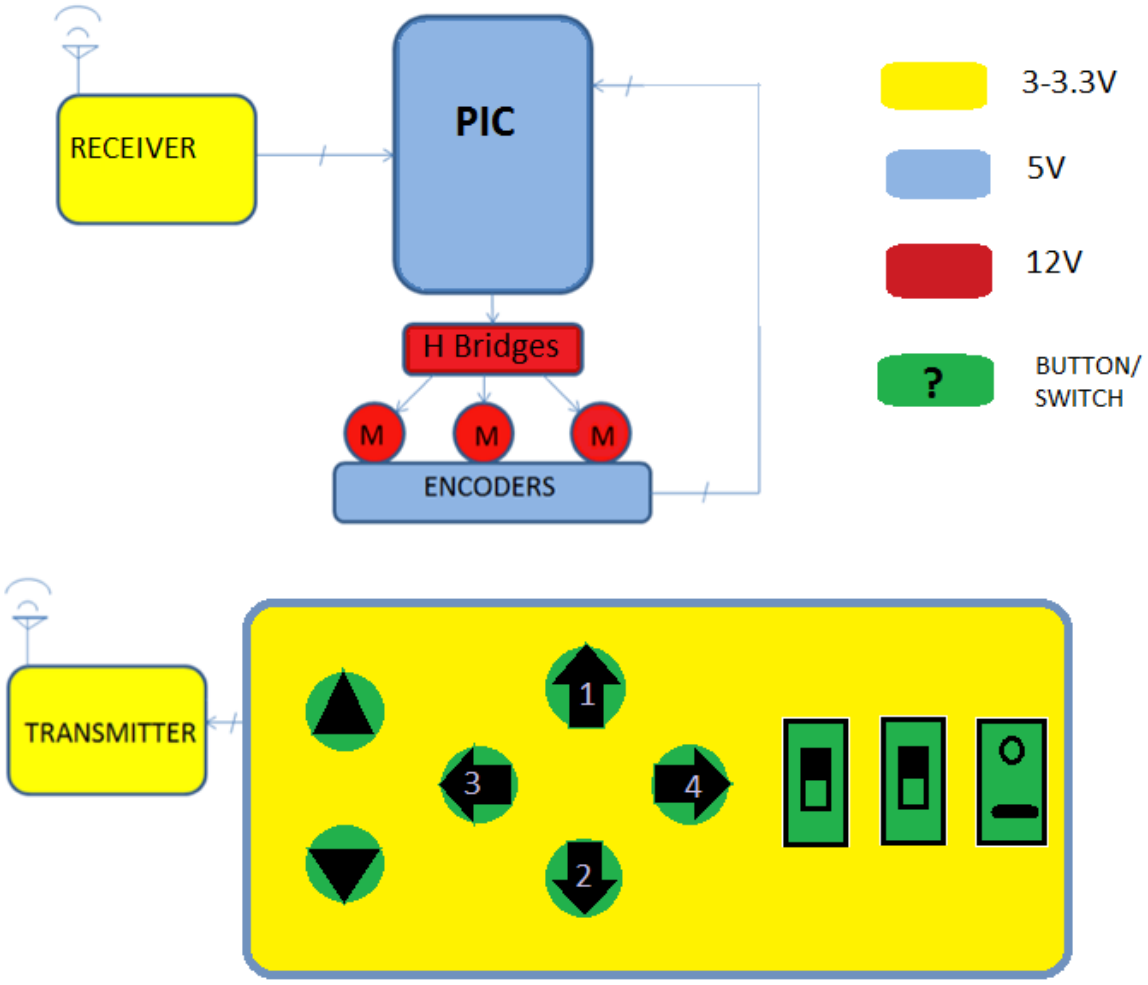

Figure 1: Block diagram

2 Design Procedure and Detail

2.1 Power supply

At first, we planned to have only two power supplies: a 12V battery to power the microphone stand and two AA batteries to power the control board. The 12V battery will be directly connected to the power pins of the H bridges to drive the corresponding motors. It will also go through a 3.3V voltage regulator and a 5V voltage regulator to power the RF modules and other logic inputs respectively. However, during testing on the breadboards, we found that the whole system became very unstable when putting the 12V voltage on the board together with 3.3V and 5V. PIC was not working properly with seemingly 12V-affected outputs even if we put diodes in front of the inputs to protect the circuit. In addition, sparks were spotted sometimes when normally operated. Thus, we decided to separate the 12V power supply with 3.3V and 5V. To do this, we added another 6V voltage supply to the system so that the 12V battery went directly to the power pins of the H bridges without connecting to the breadboard or the PCB later on. And the 6V voltage supply went through the two voltage regulators and supply power to corresponding modules.

2.1.1 12V power supply

We got the 12V battery directly from the ECE electronic service shop. It is a 12V 35AH rechargeable battery made by Xtreme.

2.1.2 6V and 3V power supply

The 6V and 3V power supply are four AA batteries in series and two AA batteries in series respectively. We used AA batteries manufactured by Energizer. They are rated 2.85 AH. Additionally, because the RF modules we used do not have internal voltage regulators, they require clean, well-regulated voltage supply. It is indicated in the datasheet that the maximum voltage ripple should not exceed 30mV. The 3V battery directly supplies the power of the transmitter module on the control board, so we need to make sure it meets the requirement. As shown in the simulation in Figure 2, the maximum peak-to-peak voltage for the 3V battery is 21.9mV, which satisfies the requirement.

Figure 2: 3V test

2.1.3 Voltage Regulators

We used two voltage regulators: 3.3V and 5V. They are both manufactured by Fairchild Semiconductor. The output from the 3.3V regulator supplies the receiver module. So, similarly, it cannot have a ripple voltage more than 30mV. Shown in Figure 3 is a simulation of the output from the 3.3V voltage regulator. We can see that the maximum peak-to-peak voltage is 16.3mV, which satisfies the requirement.

Figure 3: 3.3V test

2.2 RF Modules

We chose to use the RF modules manufactured by Linx Technologies because there are related materials on the course website. Among many kinds of RF modules, we decided to use the LR series modules, which have the features of low cost and long transmitting distance. There are three different frequencies in the LR series modules: 315MHz, 418MHz, and 433MHz. Because our project will be used in a music studio, all kinds of frequencies will be present when the microphone stand is in use, we chose the highest-frequency RF module to avoid disturbance from the environment. We also checked with Mark to make sure that 433MHz frequency was not in use.

2.2.1 Transmitter Module

The transmitter module consists of a transmitter, an encoder and an antenna. As shown in Figure 4, the encoder enables us to have eight inputs from D0 to D7. The user inputs go from the encoder to the transmitter, and then are transmitted through the antenna to the receiver module.

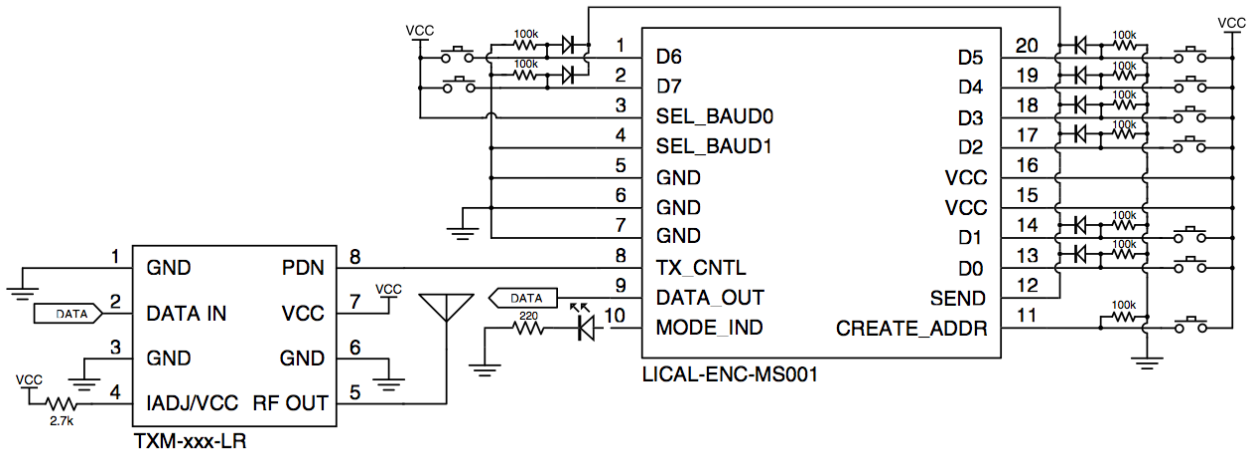

Figure 4: Transmitter module [1]

2.2.2 Receiver Module

The receiver module consists of a receiver, a decoder and an antenna. As shown in Figure 5, the receiver gets the data sent from the receiver module and transmits it to the decoder. The decoder decodes the data and outputs corresponding data from D0 to D7.

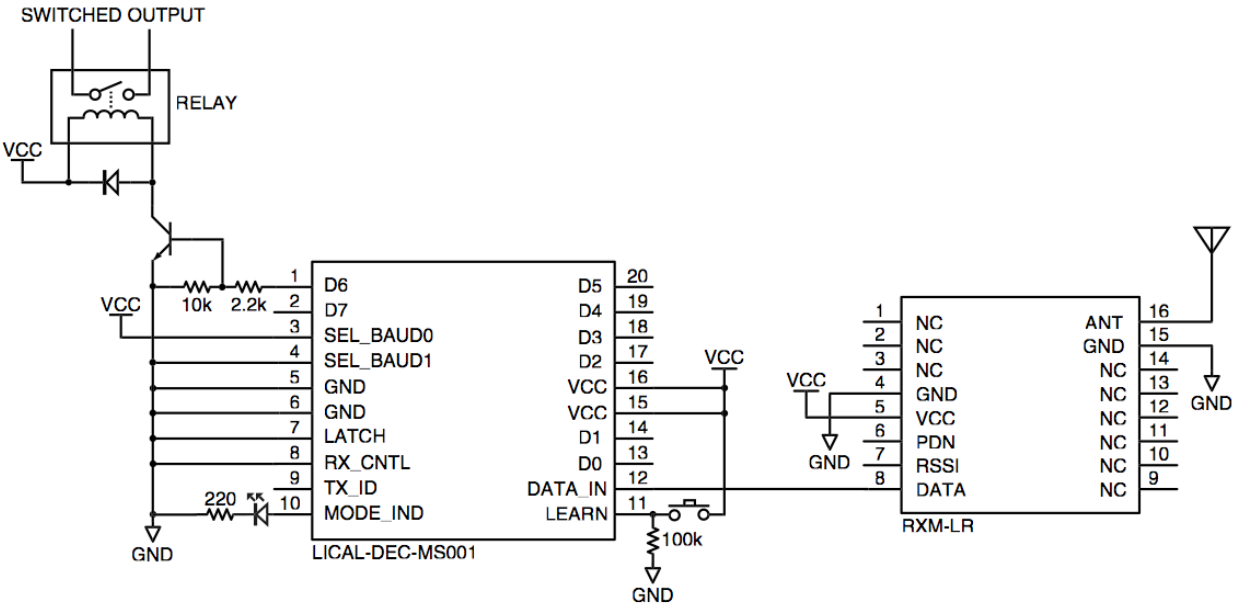

Figure 5: Receiver module [2]

2.3 Motor

Because the size of our project is very large and the materials used in the machine part are relatively heavy, the first factor that we considered for the choice of motors is the torque. Shown in Table 1 is the calculation of the total weight each motor has to drive.

Table 1: Calculations on the weight each motor has to drive

Z	Weight (lb)	Y	Weight (lb)	X	Weight (lb)
Microphone	3	Z total	3.86	Y total	9.52
mic clip	0.3	Z motor	1	Y lead screw	1.2
Lead screw nut	0.46	Z lead screw	3	Y motor	1
Fraction	0.1	Track	1.2	Lead screw nut	0.46
Total	3.86	Lead screw nut	0.46	Track	1.2
		Fraction	1	Fraction	1
		Total	9.52	Total	13.38

We measured that the gap between the motor and the setscrew is 0.267 inch and the screw diameter is 0.52 inch. The torque required to drive the machine is calculated in equation 1.

$$13.38lb \times 16\frac{oz}{lb} \times (0.267inch + 0.52inch) = 168.48oz - in \quad (1)$$

According to the datasheet of the motor without the gear head, the torque is around 8 oz-in [3]. The gear ratio need is calculated in equation 2.

$$168.48oz - in \div 8oz - in = 21.06 \quad (2)$$

From equation 2, we get the gear ration needed for our gear head motor is at least 21.06:1. The electronic service shop provides two kinds of motor that both can meet this requirement. They are 12V DC motors manufactured by Pittman with gear head ratio 65.5:1 and 127.7:1.

Another factor that we considered for the choice of our motors is the rotation speed. Because the adjustment of microphone is usually very tiny, we do not need the motor to drive the lead screw too fast. With this theory in mind, we chose the 127.7:1 gear head motor over the one with 65.5:1 gear ratio. We used three motors in total to drive the three-axis tracks.

2.4 Motor Controlling Units

Motor controlling units consists of three H bridges to control each of the motors and three mechanical motor encoders to record the motors' motion and send corresponding signals to the PIC. The H bridges we used are the only type of H bridges that the electronic service shop has. They are manufactured by Texas Instruments and rated 55V, 3A, which, in our case, are more than enough. For the motor encoders, at first, we decided to use motors with encoders pre-installed just like the ones installed on the reaction wheel pendulum used in ECE486. However, the electronic services shop do not carry any that kind of motors. It is also too expensive for us to buy them ourselves because they cost around 200 dollars per piece. Then we thought about installing optical encoders on the motors by ourselves. But still, the optical encoders cost 40 dollars per piece, which is far over our budgets. Finally, we chose to use mechanical encoders, which only cost 5 dollars per piece and have programmable output.

2.4.1 H bridges

We used three H bridges to drive the three motors respectively. Because the motors we use are 12V DC motors, the voltages supplied to the H bridges are 12V. There are three control signals: PWM, direction, and brake. They are logic signals, so they are supplied by either 5V or ground. We connected the PWM pin to high constantly because we do not need to slow the motor down or change the motor's speed. The signals of direction and brake come from the PIC.

2.4.2 Mechanical Encoders

The mechanical encoder we used has two output pins and one 5V power input pin. The two output pins output 2-bit gray code when the encoder is rotated. The encoder has 36 positions per rotation, which makes its precision 10 degrees. The motors are mounted at one side of the lead screws and the encoders are mounted at the opposite side of the lead screws. So they have the same rotation motion as the gear head side of the motors. According to the datasheet, the life time of a encoder is about 20000 rotations, which is enough for our project.

2.5 Mechanical Structure

Figure 6 shows a comparison between our original machine design and the machine we got from the ECE machine shop. They are very similar.

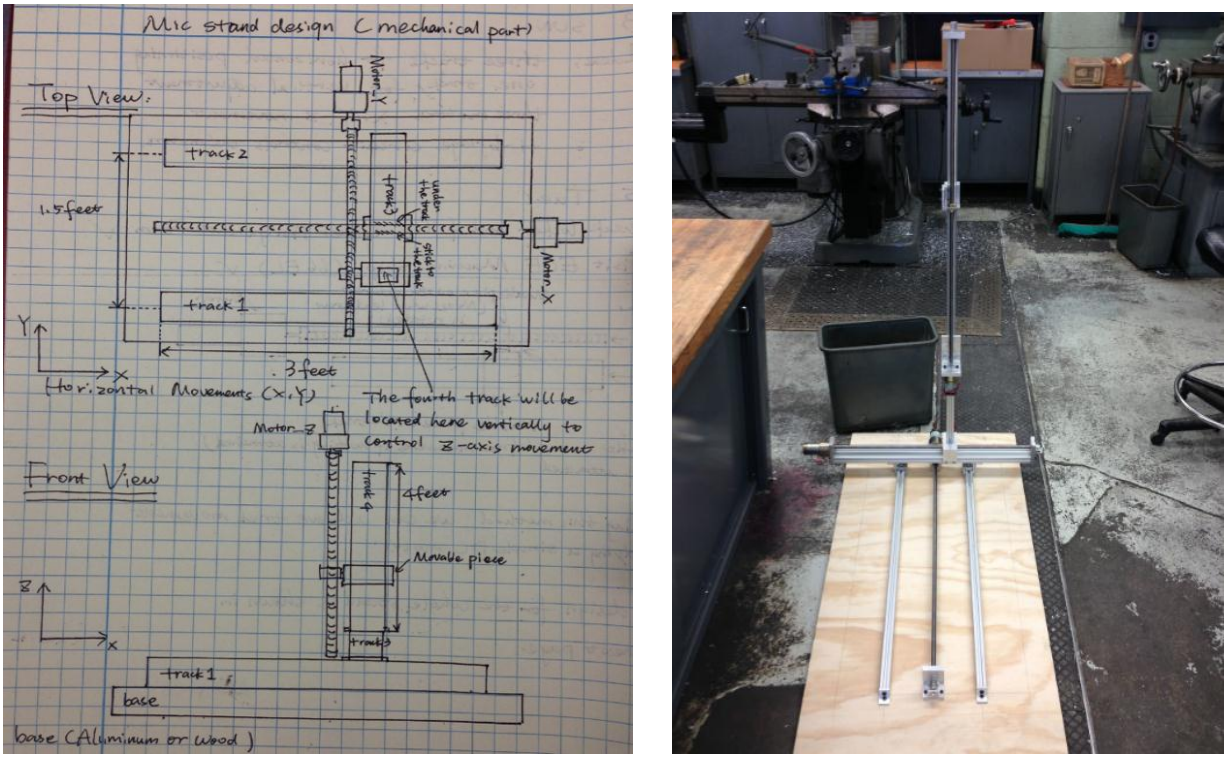

Figure 6: Our design vs. actual machine

The lead screws we chose for our machine is rated 1 inch/turn. We measured the rotation speed of our motor under no load condition to be around 60 turns/min. The resulting moving speed of the microphone is calculated in equation 3.

$$60\frac{turns}{min} \times 1\frac{in}{turn} = 60\frac{inch}{min} = 1\frac{inch}{s} = 2.54\frac{cm}{s} \quad (3)$$

From equation 3, the final moving speed will be around 2.54cm/s, which is reasonable.

2.6 PIC Software Design

PIC handles 8 inputs from the user, namely 6 direction inputs and two mode inputs, and output the proper control signal to three motors. Motor encoders also input into PIC to count rotation and generate addresses.

2.6.1 Six-direction Movement

Free movement states set the brake and direction based on user's input and count the number of rotations based on motor encoder inputs. The two lines of job need to be done at the same time and both input from user and input from encoder need to be checked continuously. For this purpose we created a state loop.

SetX and waitX both check whether the user is still pressing a button and thus decide whether or not to keep looping. SetX set the direction of motion based on user input (left/right, forward/back, up/down) and wait for encoder input 00. Counting is initialized once 00 from the encoder is recognized and state transits from setX to addX, where increment and decrement of address is handled. This state updates address by 1 and unconditionally branches to waitX. The reason is twofold. First of all we need to make sure that address updates whenever a count is recognized, regardless of the current input from the user. Second, we need to avoid multiple counts of one point and thus we force the next branch and wait. WaitX again check if the user has released the button or not while waiting for encoder to reach 11.

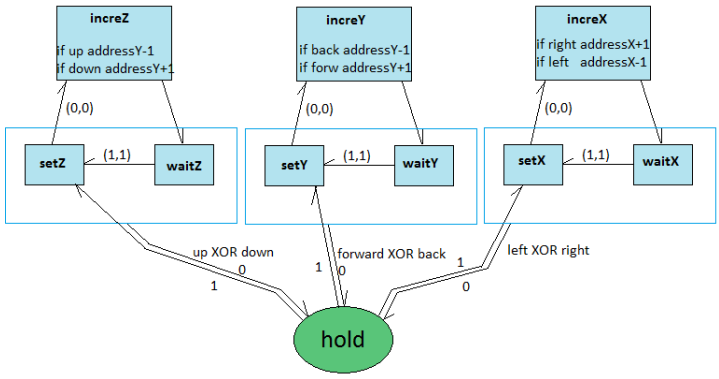

Figure 7: Six-direction movement

2.6.2 Store

Once the store switch is triggered**,** the PIC enters STORE mode. Here all motors brake and PIC waits for a store instruction. In STORE mode, the original four direction buttons now are assigned to be four

memory locations. Every press of a memory button would load the current address in three axis to its corresponding memory location for later use.

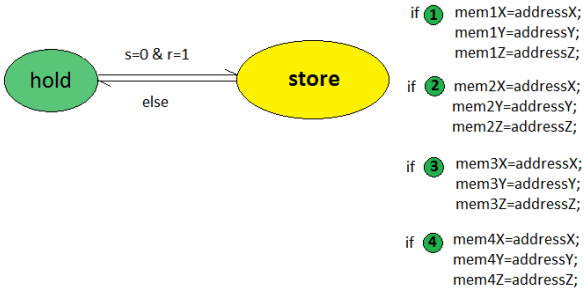

Figure 8: Store

2.6.3 Retrieve

Just like the STORE mode, the PIC enters RETRIEVE mode once the retrieve switch is high, and the four direction buttons are again functioning as four memory locations. But unlike STORE mode now the four buttons read from the memory instead of writing into the memory. Once RETRIEVE mode is triggered, all motors brake, and all free movement inputs from the user are disabled, including up and down which are not used for memory. The PIC does nothing but just waiting until a memory button is pressed, and then it enters load states and loads the address stored in the corresponding memory location into reference registers. From this point all user inputs including mode switches would not interrupt the retrieving function and PIC takes in further instructions only after the whole retrieving is done. Here a similar loop as in free movement states is adapted with comparison logic added. First the PIC compares the reference address and the current address in X direction, and branch to either an increment loop or a decrement loop based on the result. Both increment and decrement loop are similar to the free movement loop with two differences. First there are two separated loops instead of one such that the set state no longer determines the direction of motion (the direction of retrieving motion is set in the compare state). Second, instead of simply waiting for encoder 11 after address update here the PIC first compare whether the reference and current addresses are the same, and then wait for encoder 11 if they are not yet identical. Thus we call this state “check” rather than “wait”. In both compare and check states if the current address is indeed the same with the reference address, retrieve is done, at least in X direction. After finishing X retrieving PIC branch to Y retrieving with the same logic flow and then finally to Z retrieving. If all comparison in three directions turn out the same PIC goes back to wait_retrieve state and wait for further instructions. At this point user input is enabled again.

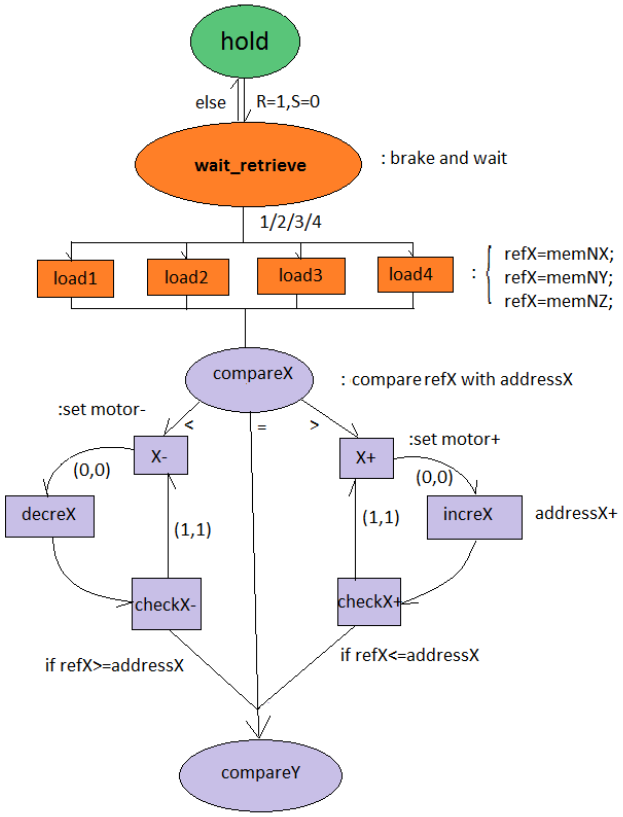

Figure 9: Retrieve

3. Design Verification

3.1 Motor and Encoder

3.1.1 Speed Verification

For the 65.5:1 gearhead motor we test its speed with no load:
Test1: 30 turn in 15 seconds

Test2: 29 turn in 15 seconds

Test3: 29 turn in 15 seconds

It roughly met the spec of 2 turns per second.

3.1.2 Final Torque Test

The final torque check is conducted after the arrival of mechanical tracks. Since the motor generates 6 times the maximum possible torque needed it smoothly drives even the heaviest X track. We pushed it with hand a little as extra resistance and the motion is barely affected.

3.1.3 Encoder

The most significant concerns about the encoder are glitches and voltage drop.

We have rotated the encoders with three levels of speed: bare hand (about 10 turn per minute), 127.7:1 gearhead motor (about 60 turns per minute) and 65.5:1 gearhead motor (about 120 turn per minute). We connected LED to the two outputs of encoder and observer the output transition. And it turns out when we rotate it when hand at low speed there are few glitches. As the speed goes up we can hardly recognize a glitch but our PIC with X direction address generation state machine provided us the capability of finding glitches. Since the state machine is designed to recognize 00 01 11 10 as a sequence any glitch in between would cause a bizarre state transition. After we ran it at both 60 and 120 turn per minutes we found that the faster it spins the more frequent happens

The second concern is that in the real mechanical structure we have to use a long wire to connect the encoder at the top of vertical axis to the PIC mounted on the PCB at the bottom. Possible voltage drop in the wires and solders might bring the 5V input below the PIC threshold for logic high. So after the final wiring of the station we measured the voltage at the PIC pin of encoder input and it turned out there is barely a drop in the voltage.

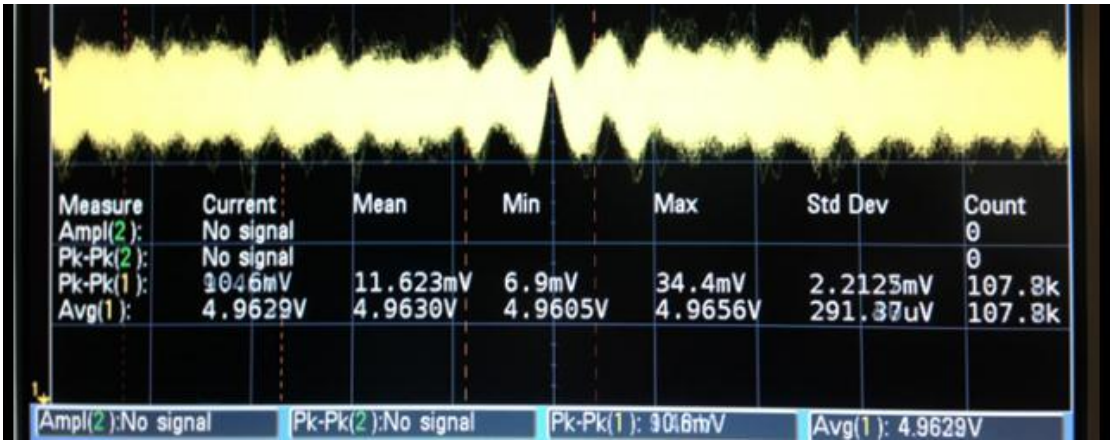

Figure 10: Encoder voltage

3.2 Wireless Communication

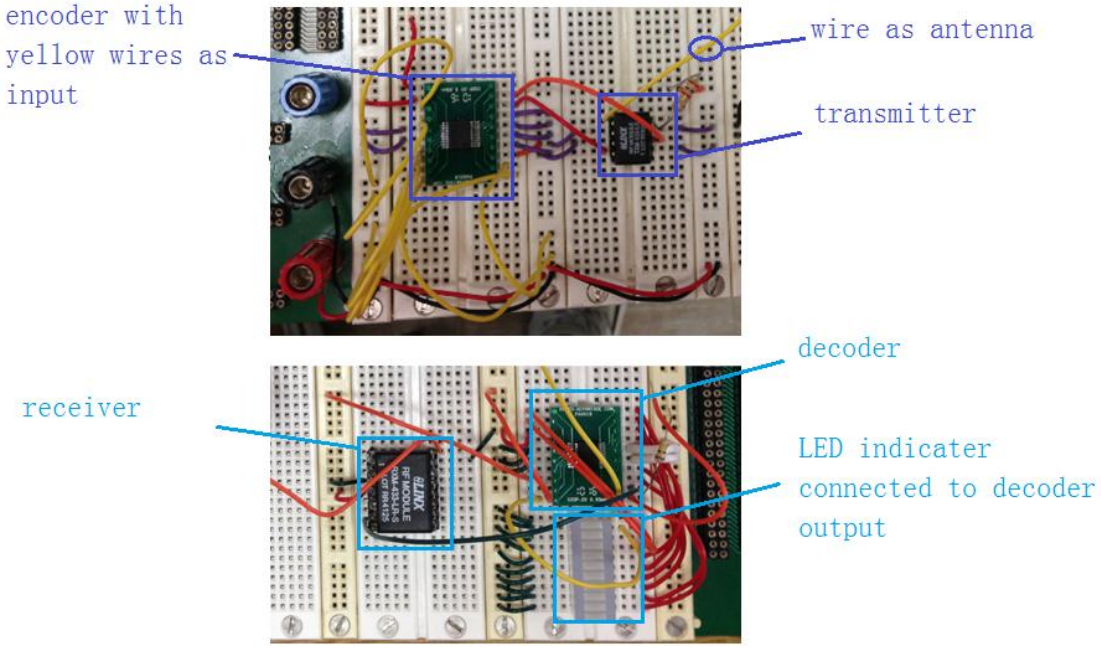

Figure 11: RF module breadboard test

3.2.1 Individual Bit Transmission

We here test all 8 bits of the en/decoder we use to make sure they all work correctly. 8 test cases are generated for 8 bits to make sure the decoder reproduces whatever the encoder sees. So we try at the encoder end 8 cases:

10000000

01000000

00100000

00010000

00001000

00000100

00000010

00000001

Decoder latch mode is grounded and encoder send is high during the test. And we see a perfect match of input and output.

3.2.2 Logic High/Low Threshold

We use 3V DC power for transmitter and encoder, 3.3V DC power for receiver and 5V DC for decoder. The concern is that the decoder is directly outputting to PIC and it has to full fill the PIC input specs. The PIC is powered also by 5V DC and it takes 0.25 Vdd + 0.8 as logic high (in theory). The decoder outputs logic high no less than Vdd-0.7 (in theory). But as we measured if we power the decoder with 3.3 V it only outputs 2.7V when mounted on bread board and 2.1V when mounted on PCB. This voltage drop may due to unexpected voltage sink in real world PCB. Moreover we found that that logic high of 2.1V into the PIC is not reliable because the PIC interprets this voltage level differently from time to time and from pin to pin. It also suffers a lot from even little turbulence in voltage. So we raised power of decoder to 5V from 3.3V, while still operate it below safety level, which is 5.5V for decoder. And now logic high from decoder rises to 3.19V and the PIC deals with this voltage level perfectly.

3.2.3 Wireless Distance and Obstacle Test

This test is carried out at the very end of our integrated tests (in section below).

We put the mic stand in the senior design lab and move around with the control board, with people and chairs in between and later on we stand outside of the room. It turned out using wire as antenna works only in small distances and on obstacles. But after putting the actual 433MHz antenna we can control the motors even outside the room with door closed.

3.3 PIC tests

We started on simple functions of the PIC and individually tested all the functions of PIC we need, and finally combined them together. After every single functionality test we modified our simple test code into the complicated version that represents the real logic we need and test it again using the same logic. The integrated tests are carried out on bread board before mechanical built-up is ready and we tested simply by tape the motor and motor encoder together and count the turns they take.

3.3.1 NO Input LED Test

This test is aimed to verify that the oscillator connected to PIC is working and the PIC can handle output high, output low and delay.

```
/*pseudo-code*/

    while(true)

{output high pin B7;delay 500ms; output low pin B7; delay 500ms;}
```

And we got a LED that flashes every half-second. Test passed.

3.3.2 Input Dependency Test

This test is to verify that PIC takes in input and translate it through predefined logic and outputs.

```
/*pseudo-code*/

    #define pin B6 as input

    while(true)

    {if (B6 is high) output high B7;

     Else if (B6 is low) output low B7;}
```

It turns out we can fully control the output by our input.

3.3.3 State Machine

State machine is the base of our logic and here we test to see if we can traverse in all the states defined and check if there are glitches between state transitions. The pseudo-code is:

```
/*pseudo-code*/

    #define state1 1

    #define state2 2

    #define state3 3

    Mystate=state1;

while(true)

{ switch (mystate):

    {case state1: if input=2, mystate=state2;

                                  else if input=3, mystate=state3;
```

```
                                    else mystate=state1;

    case state2: if input=1, mystate=state1;else mystate=state2;

    case state3: if input=1, mystate=state1;else mystate=state3;}

    switch(mystate):
            {case state1: output high B7;

     case state2: output high B6;

     case state3: output high B5;}

}
```

Tests result in smooth transition between states. Functions and outputs in a certain state are stable (LED).

3.3.4 Memory Read and Write

This is essential to our retrieving function. Since we cannot look into the PIC memory, we test memory read and write together by simple comparison.

```
/*pseudo-code*/

Char D= '0';

    Write_eeprom(memory location 10, ASCII 'm'/ 'q'/ 'e');

    Read_eeprom(memory location 10, character D);

    if D= 'm', output high B6;

    else if D< 'm' output high B7;

    else if D> 'm' output high B5;
```

B6 turns on when we write 'm' into the memory and it stays off when 'q' and 'e' are written into the memory. Instead B5 and B7 correctly indicate the result of comparison.

3.3.5 Address Generation and Comparison

Because of the length of address we use (48 bit per address), we cannot really see the actual address stored, but we can test it together by comparison.

```
/*pseudo-code*/

    while(true)

    {……
```

case increX: if clockwise address=address+1;else address=address-1;

……

case waitX: if address=0, output high B7;

else if address=1, output high B6;

else if address=2, output high B5;

……}

The count point in address are generated by the mechanical motor encoders and as we rotate it by hand in both direction, the address indicating LED switched n and off and showed the right corresponding new address. Also address updates every time passing a check point (in our case we counted 00 from motor encoder).

3.3.6 Integrated Test 1: Free Motion

This test is conducted after finishing the coding related to X free movement states and these states only. Motor is powered and controlled by PIC and H-bridge together.

Test case is:

If 'left' is high, motor spin clockwise.

If 'right' is high, motor spin counter clockwise.

If both are low, motor brakes.

Tests results in fully control of the motor.

Later on Y and Z free movement states are modified from X states and added. Same tests were applied to all directions and typos corrected.

After adding store and retrieve modes (but without retrieve functions) we need to verify that all motor brakes in these two modes. Test logic is:

Enter store mode

Input left/right, forward/back, up/down

Quit store mode

Input left/right, forward/back, up/down

Enter retrieve mode

Input left/right, forward/back, up/down

We expect that PIC ignores all the input during store and retrieve and the motors were indeed at rest with any input pressed.

3.3.7 Integrated Test 2: Store and Retrieve

This test is conducted after finishing all free movement states and X retrieving states and X retrieving only. Test case is:

……

Enter store mode and store the current address into memory location 1

Rotate left 5 turns

Enter retrieve mode and press memory location1

Expect motor to spin right 5 turns

……

Enter store mode and store the current address into memory location 1

Rotate right 20 turns

Enter retrieve mode and press memory location1

Expect motor to spin left 20 turns

It turns out 5 turns of retrieving works well in both left and right directions, but 20 turns retrieving results in disaster. After storing position and rotated left 20 turns, the motor tried to retrieve by still rotating left. We addressed the problem to be an overflow of address generation and thus we changed we address type from integer to long integer. After the modification 20 turns worked.

Later on same method was applied for Y and Z retrieving and again typos were checked and corrected.

4. Costs

4.1 Parts

Table for parts costs is shown below.

Table 2: Parts costs

Part	Manufacturer	Quantity	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
12V Battery	Xtreme	1	24.95	24.95	0
AA Batteries	Energized	6	1.29	6.99	6.99
5V voltage regulator	Fairchild Semiconductor	1	0.99	0.99	0.99
3.3V voltage regulator	Fairchild Semiconductor	1	0.99	0.99	0.99
Motor	Pittman	3	19.99	18.99	59.97
H bridge	Texas Instruments	3	1.99	1.79	5.97
Mechanical Encoder	Grayhill	3	5.12	4.97	15.36
RF transmitter	Linx	1	6	6	6
RF encoder	Linx	1	3	3	3
RF receiver	Linx	1	6	6	6
RF decoder	Linx	1	3	3	3
Antenna	Linx	2	7.45	7.45	14.9
PIC Micro-controller	Microchip	1	7.85	7.33	7.85
Pushbutton	N/A	6	0.49	0.45	2.7
Toggle switch	N/A	3	0.88	0.82	2.64
Aluminum tracks	N/A	4	10	10	40
Lead screw	N/A	3	5.95	3	17.85
Total actual cost					**194.21**

4.2 Labor

Table for labor costs is shown below.

Table 3: Labor costs

Team Member	Ideal Salary/Hour ($)	Hours Spent	Cost (Multiplied by 2.5) ($)
Kai Jiang	40	200	20000
Weihong Yuan	40	200	20000
Total	40000		

Combining Table 2 and Table 3, the estimated total cost for our project is $40194.21.

5. Conclusion

5.1 Accomplishments

We have fulfilled all the requirements of this project.

5.2 Uncertainties

There are two sources of error

The first is the theoretical error which takes place because we initialize a counting only when the motor encoder reaches 00. But it might happen that the motor encoder is at rest at 11, 01 and 10 with the same possibility before counting. So the encoder state has to be passed before counting generate the theoretical error whose maximum value is 3. Since we have 36 counts per rotation and 1 inch per rotation, the error is given by equation 4:

$$\frac{3}{36} \times 1inch = 0.08inch = 0.2116cm \tag{4}$$

This error does not accumulate and it happens only once when trying to initialize a count.

The second error is given by encoder glitches, because the mechanical motor encoders we used are too cheap to be accurate. Some of our tests show that when the encoder rotates from 11 to 10, a 00 might take place in between, and some other glitches are also expected to happen. We have discussed the relationship between rotation speed and glitches above. This kind of error is unpredictable and once it happens it causes mis-count or over-count. We picked the longest track (X track) and ran retrieving from one end to the other 10 times. It turned out than the maximum error we were able to get is 0.11 inches =0.28 cm, slightly higher than the theoretical value because it measured the combined effect of both. Maximum percentage error is given by equation 5:

$$\frac{0.11inch}{3feet} \times \frac{1feet}{12inch} < 1\% \tag{5}$$

5.3 Ethical considerations

We do not think that our project has any conflict with the IEEE Code of Ethics. The ninth code of the IEEE Code of Ethics states that “to avoid injuring others, their property, reputation, or employment by false or malicious action.” The device we are designing will be remotely controlled from the listening area of the studio. The experience of visiting Pogo Studio in downtown Champaign makes us aware of the fact that it is very hard to see the microphone stand’s movements from the listening area because of the thick sound-proof wall between the listening area and the recording area. So, it is possible that the microphone’s position is over-adjusted and hit other musical instruments causing unexpected losses. To solve this problem, we will simply install a camera pointing towards the microphone monitoring the movements of the stand.

5.4 Future work

There are improvements that can be made to this project.

First of all, as Professor Singer pointed out, it is possible that somebody's hair get stuck by the track and this might cause injure to human. The solution is to put an emergency stop button on both the control pad and the mic stand to stop the motors in dangerous conditions.

Second, optical encoders could be used to replace the mechanical encoders at the cost of a higher budget. The encoders we used can work for only 12000 turns but optical encoders have much longer operation life, and thus we do not need to replace them from time to time. Also, optical encoders are more accurate. By using optical encoders we hopefully can eliminate the unpredictable error discussed above. However the optical one on the market costs at least $40 each while the mechanical ones we used costs only $5 each.

Moreover, it would be timesaving to move and retrieve in three directions together. We have not figure out a way of doing that by one PIC which performs only one "while loop" at the same time. But it is possible to do this by adding two more PICs, one controlling each motor.

References

[1] Linx Technologies LR Series transmitter module, web page. Available at: https://www.linxtechnologies.com/resources/data-guides/txm-xxx-lr.pdf Revised November 2011.

[2] Linx Technologies LR Series receiver module, web page. Available at: https://www.linxtechnologies.com/resources/data-guides/rxm-xxx-lr.pdf Revised November 2011.

[3] Pittman motors datasheet, web page, Available at: http://www.gearseds.com/standard_motor.html

Appendix A Requirement and Verification Table

Table 1 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Motor/Encoder a) Z motor torque>3.86lb-0.787in b) Y motor torque>9.52lb-0.787in c) X motor torque>13.38lb-0.787in d) encoder output>1.8V with final setup	1. a) Z motor torque over 6 times larger b) Y motor torque over 6 times larger c) X motor torque over 6 times larger d) encoder takes in 5V outputs back 4.9V	Y
2. wireless connection a) decoder should reproduce exactly the same 8-bit signal of encoder input b) decoder output voltage should be higher than PIC logic high level (1.8V) c) receiver should receive signal at least outside 10m range and with obstacles(people, glass, door…) inbetween	2. a)every single bit of transition should be tested with decoder latch mode pin low b) mount decoder on final PCB powered by 5VDC , test decoder voltage high level (3.1V) c) after PCD soldering take control board out of design lab and try to control the mic stand	Y
3. PIC test states/input/output a) independent output test b) dependent output test c) state transits based on input and state outputs accordingly d) read/write/address generation/comparison	a) PIC controls LED with 500ms time period b) PIC controls LED upon input c) Output A3 A2 as state indicator and output A1 A0 as direction and brake instruction. Test if input could fully control direction and brake bits. Also state should transit properly. Then connect to motor and input should drive motor in related direction. While no input motor brakes. d) Four LEDs corresponds to address being 0 1 2 3. Address should be updated immediately after passing a count point.	Y
4. Integrated test a) free movements b) store/retrieve mode	a) all six directions of movement should be controlled by control board b) no free movements in store/retrieve modes. Retrieve previous position	Y