



Resource Center for Persons with Disabilities
Maximizing Ability & Opportunity

Doug's Kitchen Robot

Sponsored by

Michigan State University Resource Center for Persons with Disabilities
Stephen Blosser

Michigan State University

ECE 480: Senior Design

Design Team 3

Thomas Manner
Daniel Phan
Ali Alsatarwah
Ka Kei Yeung
Facilitator: Lixin Dong

Executive Summary.....	5
Acknowledgments.....	5
Chapter 1: Introduction and Background	6
Chapter 2: Exploring the Solution	8
Fast Diagram	8
Figure 1: Fast Diagram	8
Critical Customer Requirements.....	9
Concept I	9
Concept II	9
Concept III	9
Figure 2: Concept I – Triple Joint Robotic Arm	10
Figure 3: Concept II – Wall Mounted Cartesian Robotic Arm.....	10
Figure 4: Concept III – Wall Mounted Cartesian Robotic Arm with Rotating Arm	10
Figure 5: Design Criterion Rankings Diagram	11
Initial Budget	12
Figure 6: Initial Budget.....	12
Gantt Chart.....	12
Chapter 3.....	13
Hardware Design Efforts	13
Figure 7 XBee Explorer	13
Figure 8 Servo Arm Duty Cycle vs. Position	14
Figure 9 Pololu High Power Motor Driver 18v15.....	15
Figure 10 Left: Motor Chopper & sensors. Right: Graphical Illustration	16
Figure 11 GP1A73A Schematic ⁵	17
Figure 12 Modified Window Motor with added casing, chopper and Photo Interrupter....	17
Figure 13 Equivalent Schematic for each joystick direction	18
Figure 14 Seeed Studio Joysticks and Illustration	19
Figure 15 Controller Console (Top View)	19
Figure 16 Demonstration System	20

Figure 17 Receiver Box.....	21
Figure 18 Display Box.....	21
Software Implementation.....	22
X-CTU.....	22
Arduino Development Environment.....	23
Arduino Code	23
Controller	23
Receiver.....	24
Chapter 4: Testing and Functionality.....	25
Wireless Testing.....	25
Pulse Width Modulation Testing	26
Analog Input Testing	27
Interrupt Testing.....	27
First Motor Driver Tests	28
Photo Interrupter Testing	28
Power Supply and Window Motor Testing.....	28
Demonstration Design and Testing	29
Figure 19 - Motor Feedback: Clockwise.....	30
Figure 20 - Motor Feedback: Counter-Clockwise	31
Chapter 5: Conclusion.....	32
Final Cost.....	34
Appendix 1 – Technical Roles	35
Thomas Manner.....	35
Daniel Phan	36
Ali Alsatarwah	37
Ka Kei Yeung.....	38
Appendix 2 – Gantt Chart	39
Tasks and Task Assignments	39
Project Timeline	40

Appendix 3 – Controller Arduino Code	41
Appendix 4 – Receiver Arduino Code	43
Appendix 5 – XBee Modem Configuration Files	48
Modem 0000: Programmer	48
Modem 0001: Controller	49
Modem 0002: Receiver	50
Appendix 6 – Photo Interrupter Data Sheet	51
Appendix 7 – Arduino Fio Schematic	56
Appendix 8 – System Block Diagram and Schematic	57

Executive Summary

The goal of our project is to design and construct a wall mounted robotic arm to help a quadriplegic person become more independent. The arm is intended to assist him in cooking and lifting heavy items such as a pot of boiling water. The design goals of this arm involved creating a wireless joystick interface that would allow the user to control the speed and direction of the motors in the robotic arm. The controller interface would then be programmed to give the robotic arm Cartesian movements. Our team was able to attain the goal of creating a wireless joystick interface that could be used to control the speed and direction of a motor. The interface consisted of three joysticks connected to an Arduino FIO board that generated a signal to control the duty cycle of a pulse width modulated signal. That signal is then transmitted wirelessly to another board where a PWM signal is generated with the corresponding duty cycle. Our team also provided sensing feedback system using two IR interrupters to provide the signals that could be used to determine the speed and direction of the robotic arm. The construction and operation of the gripper was also achieved using the wireless joystick controller. The impact of the work our team accomplished this semester will significantly simplify the direction the next team is going to need to take to complete this project.

Acknowledgments

Team 3 would like to thank Stephen Blosser, Dr. Shanblatt, Dr. Dong and the entire ECE faculty. Their assistances and contributions help our team to achieve the goals of the project by creating small scale prototype for the Doug's Kitchen Robot

Chapter 1: Introduction and Background

The project goals set for Design Team 3 were to design and build a robotic arm that would assist a person with quadriplegia in cooking his own meals independently. To accomplish such a task, our team had to first determine what features and functions the robotic arm would have. The robotic arm would be required to lift and transport heavy kitchenware and pots of water weighing up to 30 pounds from the sink to the stove. It would also be required to be versatile enough to pick up smaller items that would be present in a kitchen environment and aid in the process of cooking a meal. To achieve the needed versatility, our team would be required to develop a gripping system that could achieve the stated goals and implement a stirring feature. To control these features and the movements of the robotic arm, our team would also be required to design a controller interface. The controller interface would need to feature a three joystick layout for manipulating movements, gripping, and stirring features of the robotic arm. Those joysticks will be able to vary the speed at which the arm is moving. Other features and functions will be added for overall safety and will be discussed in later chapters. After reviewing many of the required features and functions, our team next had to develop a design of the mechanical structure for the robotic arm. Research had to be conducted towards possible designs and the required motors to handle the payload. Our team would need to develop a system for controlling those motors and consider the signals that would control the speed and direction.

The intention of this project is to provide Doug, who is quadriplegic, with a robotic arm that would be installed in his home. The robotic arm would assist Doug in overcoming some of his physical limitations as he prepares meals for himself. To design a robotic arm that could accomplish this, our team would first need to consider the Universal Design Principle. This principle requires us to make a conscious effort to make the robotic arm more accessible to physically handicapped people. After extensive research into people with quadriplegia and a meeting with Doug, our group learned the physical limitations that we must accommodate in order to make this robotic arm more accessible. Those physical limitations include being bound to a wheelchair, limited arm and hand dexterity, and generating enough strength to lift heavy wares. Features were added to both the robotic arm and the controller interface to accommodate and overcome those physical limitations.

The main goal of the project is to design and build a robotic arm that is intended to assist a quadriplegic person in the kitchen. To accomplish this goal, objectives must be completed within the project. One of these objectives is to accommodate the limited physical dexterity of a quadriplegic person's hands in the design of the controller. To accomplish this objective, we will design a layout where the joysticks are spread out and

programmed so only one joystick can be active at any one time. This will prevent unintended movement of the robotic arm if the user accidentally bumps into another joystick. Another objective is to design a wireless controller interface. Since a quadriplegic person is bound to a wheel chair with limited mobility, a wireless controller provided a safety feature in which our user is not tethered to the counter. The other objective of the project is DC motor control through the use of H-bridges and providing a feedback sensing system. Those signals will be processed in the microcontroller board that we chose. This will provide a way to control the motors that are used to move the robotic arm. Our team will also need to find a gripping system for holding heavy wares and pots of water.

Commercially available kitchen robotic arms tend to be smaller in size, have automated movements, a small payload, and are very specialized. These robotic arms are generally designed to be feeding robots and stirring robots. They can be programmed to remember positions and go to those positions after the designated period of time. These smaller models tend to cost anywhere from \$200-\$6000 and lack versatility. The larger models that could be adapted to this purpose would cost more than \$10,000. This price would not include the service cost of maintenance and calibration needed to maintain the robotic arm. Most robotic arms available in the marketplace are also floor mounted and the larger ones tend to weigh over a ton. There does not seem to be a market for commercially affordable robotic arm capable of handling the payload required by our project description.

The robotic arm our team is designing and building will take a very different approach from what is considered conventional today. Our robotic arm will be wall mounted with a track along the wall instead of floor mounted. This will increase the amount of area that can be covered by the robotic arm. Our robotic arm will consist of an X track, Z track, rotational joint that will move a straight arm, and a rotational gripper. This is different from the traditional mechanical structures of the joint robotic arm and Cartesian robotic arm. The overall structure of the robotic arm will be much lighter compared to its large commercial counterpart. The robotic arm we are designing will also be manually controlled and designed to complete multiple tasks unlike the smaller robotic arm available. The robotic arm will be successful due to its versatility, ability to handle the payload required in the kitchen, and the work space it can cover.

Assuming our project is a success; the design of the robotic arm can be duplicated and modified to assist people with different kinds of handicaps. This product will ease the lives of handicapped people and allow them to live their lives more independently. However, a large scale production of these robotic arms is highly unlikely due to the

maintenance required for its upkeep. Nevertheless, it is possible to build a robotic arm while spending considerably less than the cost of commercially offered robotic arms. Therefore, one of our main design perspectives is to provide a robotic arm that is both versatile and affordable.

Chapter 2: Exploring the Solution

Fast Diagram

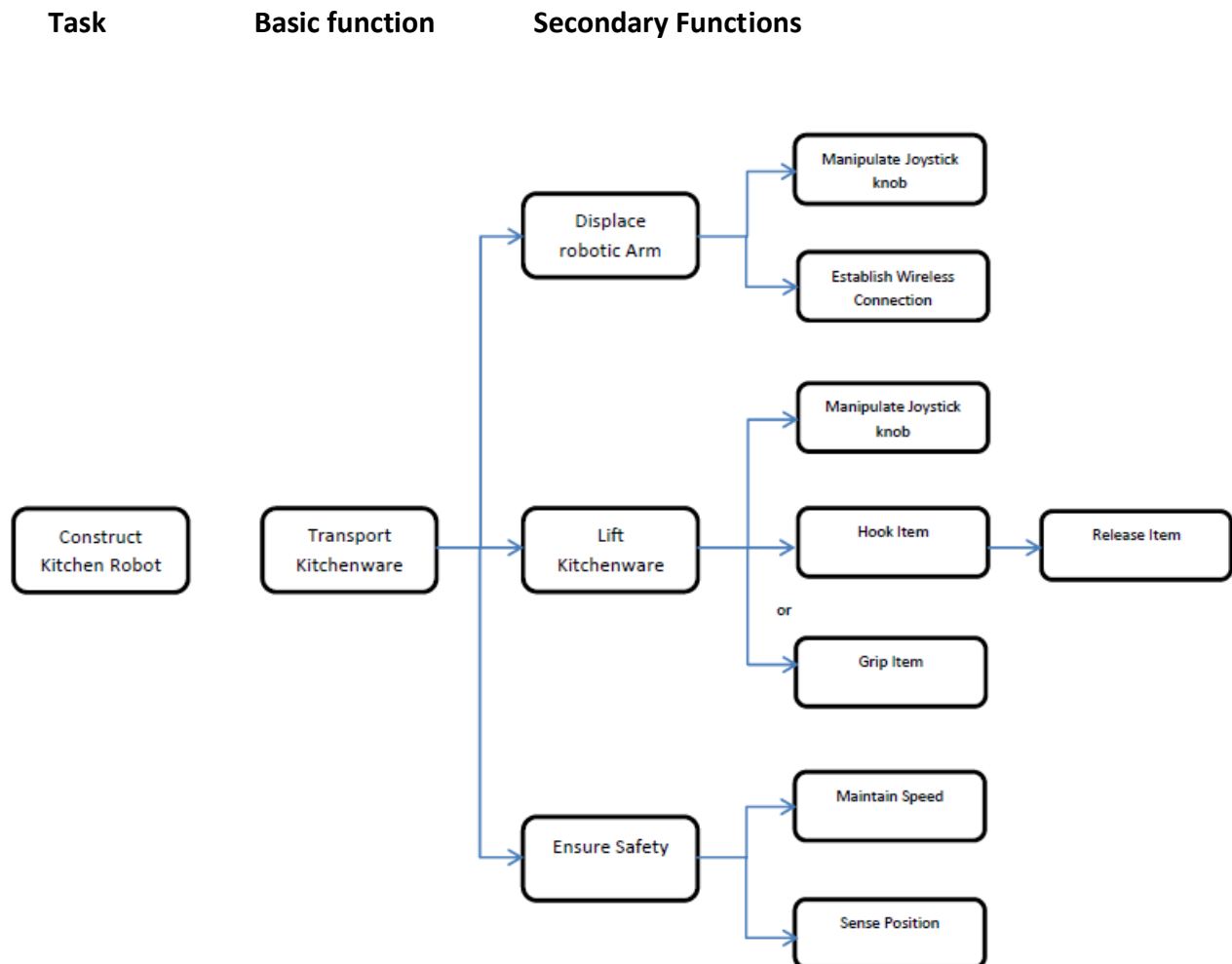


Figure 1: Fast Diagram

The fast diagram was developed as a way of visualizing the needs and final objectives that would be required of the robotic arm and joystick interface. By defining those needs and functions, our team was able to design a robotic arm that fulfilled those goals. The paths of the fast diagram outline our thought process in developing each function and the plans to accomplish those functions. Throughout the design process,

the fast diagram would provide a reminder of the goals that our team needed to keep in mind when developing the robotic arm and joystick interface.

Critical Customer Requirements

To design the robotic arm, our team was required to determine what customer necessities needed to be satisfied. This dictated that an interview be conducted with both our sponsor and Doug, the person who is to receive the finished robotic arm. In these meetings our team determined what features and functions are needed to fulfill those necessities. Those features and functions include a wireless controller accessible to Doug and a mechanical arm that is capable of lifting a pot of water or other heavy kitchenware. In discussions with our sponsor, our team originally developed three different concepts for the mechanical structures for a robotic arm.

Concept I (shown in Figure 2) is a design for a robotic arm with 3 joints and a gripper that will be situated on Doug's wheelchair and would be removable. The base joint is capable of a half-circle rotation on the horizontal plane and a half-circle rotation on the vertical plane. The middle joint is capable of approximately a half-circle rotation (with limitations due to arm thickness). The gripper joint is capable of a full-circle rotation. The controller for the robotic arm would control movement at each joint.

Concept II (shown in Figure 3) is a designed to be a Cartesian robotic arm that will be wall mounted in the kitchen. There will be three linear track actuators for the X, Y, and Z track. The gripper is attached to the bottom of Z arm and is capable of one full-circle rotation. This arm will be wirelessly controlled with a controller console with three joysticks. The controller console is mobile and could be placed on Doug's wheelchair.

Concept III (shown in Figure 4) is a Cartesian Robotic arm that will be wall mounted in the kitchen. This prototype will have two axes (X and Z). These axes will both be situated against the wall. In order to retrieve items, the arm will rotate about the Z axis for a maximum of 180 degrees. A gripper is attached to one side of the bottom of the arm end and hooks are attached to the other side. The gripper is capable of full-circle rotation and is designed to handle a lighter payload. The hooks will be able to handle a heavier payload. The controller design is similar to that of Concept II.

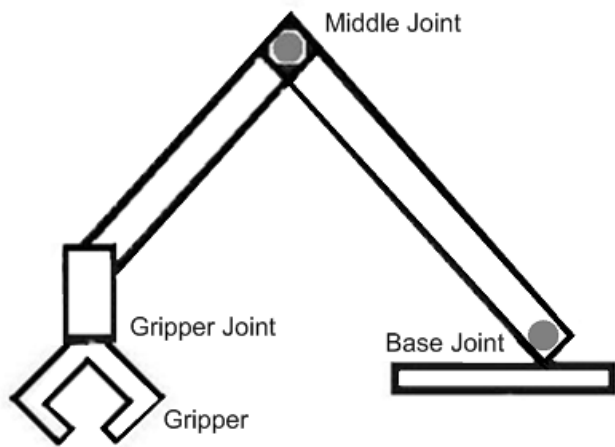


Figure 2: Concept I - Triple Joint Robotic Arm

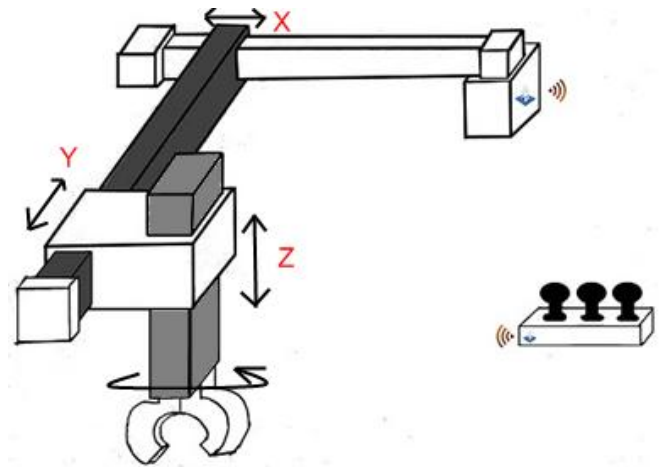


Figure 3: Concept II - Wall Mounted Cartesian Robotic Arm

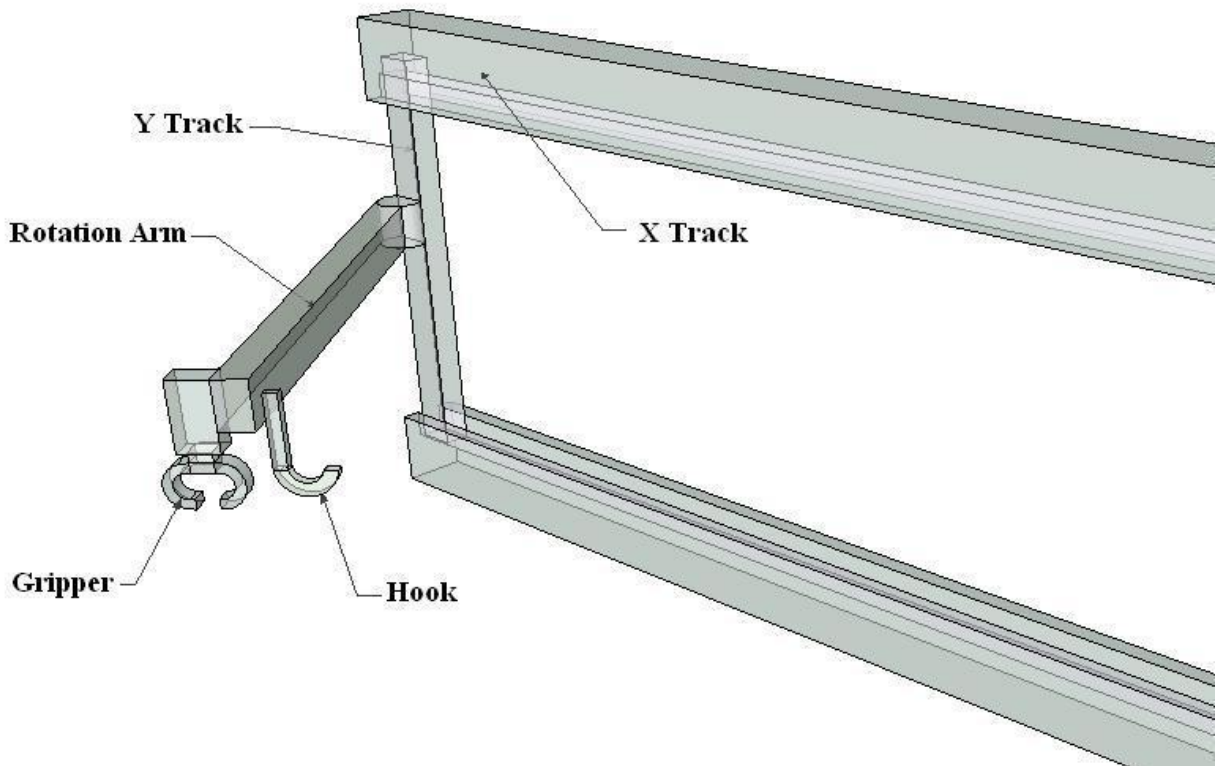


Figure 4: Concept III - Wall Mounted Cartesian Robotic Arm with Rotating Arm

For each concept, we explored and ranked the feasibility of each structure's ability to fulfill the requirements set by the sponsor and Doug. To organize these rankings our team applied the House of Quality technique. In this technique, we would provide design criterion and listed the rankings of each criterion's importance according to our team. Our team used this technique to form a decision matrix to help find the most feasible concept. We then rank the feasibility that each of concepts has to accomplish each of the design criteria. Below is the design criterion diagram that our team developed to determine the feasibility of the design:

DESIGN CRITERION	Importance		Wall Mounted Cartesian Arm	
		Concept I Triple Joint Arm	Concept II (3 Axes)	Concept III (2 Axes 1 Rotation)
Aesthetic	2	4	3	2
Coding Complexity	2	2	5	4
Durability	4	2	3	4
Flexibility and Orientation	3	5	4	4
Payload	4	2	5	5
Power Consumption	4	3	2	4
Retractability	4	4	2	5
Safety	5	2	4	5
Size	4	5	2	3
Sturdiness	5	2	5	4
User Friendliness	4	4	5	4
Upgradability	2	2	5	4
	Total:	131	159	177

Figure 5: Design Criterion Rankings Diagram

The chart shows how our team ranked each concept according to each design criterion. The importance rating for a criterion is then multiplied by its corresponding feasibility rating for each concept. This is done for each criterion listed and the scores are summed for each concept. Those totaled score are listed in the last row of the table. For the three concepts, Concept III scored the highest according to the design criterion rankings followed by Concept II and Concept I.

For the controller interface, our team decided to implement a wireless three joystick controller design. To determine the design layout of the joysticks and joystick knob, our team scheduled a meeting with Doug. In this meeting, our team interviewed Doug on how we could make the design more accessible to him. Our team decided that the knobs of the joysticks would be spaced on the controller with the top joystick on the left, middle joystick on the right, and bottom joystick on the left. The joystick knobs

would be large enough for him to manipulate. For the programming of the joysticks, our team decided to allow only one joystick to be active at one time.

Initial Budget

After designing the robotic arm and determining the basic features for the joystick interface, our team had to research what parts would be needed to realize our design. Initially, we developed a budget to reflect the parts our team thought to implement in the project. For the robotic arm, we thought to use linear track actuators as our X and Z track, robot hand with servo motor, arm rotor motor, and servo motor. For the wireless joystick controller, our team decided to implement two Arduino FIO microcontrollers with wireless programmer and three analog joysticks. The table below shows the initial cost estimates for the parts we originally chose for the project.

	Unit Price	Quantity	Item Total
Arduino Fio	\$25	2	\$50
XBee Radio	\$23	3	\$69
Linear Actuator (40" and 20")	\$130	2	\$260
Standard Servo	\$17.99	1	\$17.99
Force Sensing Resistor	\$5.00	2	\$10.00
Joysticks	\$25	3	\$75
Miscellaneous Components	\$40	1	\$40
Hitec Spline Metal Servo Horn (Tapped)	\$3.25	1	\$3.25
Force Sensing Resistor	\$5.00	2	\$10.00
Standard Servo	\$17.99	1	\$17.99
Robot Hand – A signal (no servos)	\$17.89	1	\$17.89
USB cable A/male to B/male Model	\$9.01	2	\$9.01
Arm Rotor	\$40	1	\$40
Total:			620.31

Figure 6: Initial Budget

The initial budget shown in figure 6 above is just an early estimation of the amount of money our project will cost to realize. Throughout the semester our team conducted more research to find more efficient designs and affordable parts so that the project would come closer to coming under budget. The final cost for everything in the project will be shown in a table in Chapter 5 of this report.

Gantt Chart

The Gantt chart was developed as a part of the project management plan in which our team created a time line for the completion of the project. In the development of the

Gantt chart, our team listed the tasks that we deemed as crucial to the completion of the project. Each task was then assigned to members of the group to complete during the time allotment specified by the timeline. The Gantt chart acted as a loose guide to help direct our team in the completion of the project. The Gantt chart our team devised (listed in Appendix 2), shows the tasks we deemed important and team member assignments to each of those tasks. Slight changes were made to the Gantt chart since week 4, so that our team might develop a demo to show at Design Day.

Chapter 3

During our preliminary designing meetings with our sponsor, Mr. Blosser, we discussed our options to producing the mechanical portion of the design. While we have not yet finalized our decision on this, the team focused on creating the wireless control interface, gripper motion control and a feedback system to detect and update position of the arm.

Hardware Design Efforts

Our overall design was controlled by two Arduino Fio controller boards. The Arduino Fio is based on the ATmega328P and is intended for wireless applications through XBee Radios. To program these boards, we would upload sketches (i.e. codes for programming Arduino boards) using an XBee adaptor (XBee Explorer Regulated) and an additional XBee radio which acts as a programming tool.

Before mounting the XBee radio on the Explorer as a programmer (**0000**), a jumper needs to be soldered as shown in Figure 1. The programmer would then require configuration using a free software X-CTU. Similarly, our receiver XBee (**0002**) and controller XBee (**0001**) would need to be configured using the Explorer so each of them is restricted to specific functionalities (this will be discussed in the software section).

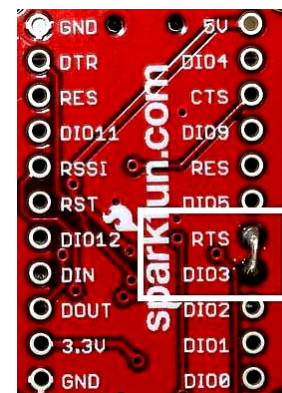


Figure 7 XBee Explorer

In our design, receiver XBee/Fio terminal (**0002**) would be used as the central motor control. The Digital I/O pins 3, 5, 6, 9, 10 and 11 of the Fio could be configured to provide 8bit PWM signal. The gripper's grip position is controlled by a servo motor whose rotation angle could be determined by the **duty cycle** of the PWM input. After a simple assembly of the gripper and servo following a guide found in lynxmotion.com¹, we tested the specification of the gripper motion. Since the datasheet of the servo motor vaguely stated the relationship between Servo Arm angular position and the period of its

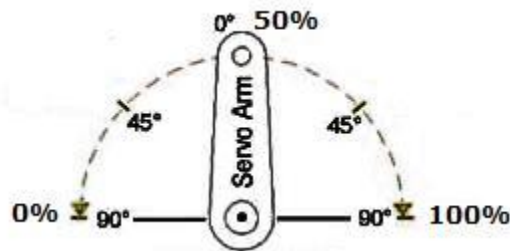


Figure 8 Servo Arm Duty Cycle vs.

input signal (without specifying the signal type), we tested the motor to retrieve more specific data. Using a function generator, we provided a square wave output of 250 Hz and varied its duty cycle from 0% to 100%. We found the corresponding duty cycle and servo arm position as shown in figure 2. We concluded that the motor detects consecutive rising edges (or falling edges) and calculates the “HIGH” (or “LOW”) duration to determine its angular position. Since it was possible to manipulate duty cycles with a steady frequency in the Arduino codes, we used this method to design our gripper control. During one of our testing stages, the servo was set to a closed grip position (0°/50% Duty Cycle) for 10 minutes during and it heated up significantly. The reading of on current monitor on our power supply reported a draw of over 600mA. We concluded that the servo was not functioning correctly and this temperature was definitely not safe for kitchen operation. While safety is one of our major design prospects, this accidental discovery led us to re-use two similar servos that we found previously. These servos were tested for functionality and the one that did not have unreasonably high current draw was the HSR5995TG ([datasheet of similar model](#)²). After we replaced the servo into the gripper assembly, we had to determine the two boundaries of the gripper's open and close position. Its closed position was 25% duty cycle without stalling and its open position was 60%.

While the bi-directional X-axis, Z-axis, gripper's rotation and rotational arm are driven using window motors (for heavy payload), they would require a circuit capable of switching their directions while handling a relatively high a mount of current. We

consulted our teaching assistant, Steve Zajac for a possible solution to this challenge and he suggested us to use a power Op-Amp. We explored this option and purchased a High-Voltage, High-Current Op-Amp from Texas Instruments (OPA549S3). This option is feasible but it would require additional circuit components to regulate power supply, as well as an IC to control directions of the current flow (thus the rotational direction of the motors). While discussing this issue during one of our weekly meetings with the sponsor, he insisted that the circuit design was too complex and would not align with one of our design goals – simplicity. The team and the sponsor agreed that it was best to use commercially available H-Bridge Motor Drivers instead of the Op-Amp. To meet our budget limit, the team researched for the best option and acquired several relatively cheap H-Bridge motor drivers (from Pololu.com⁴). These motors drivers are rated at 18

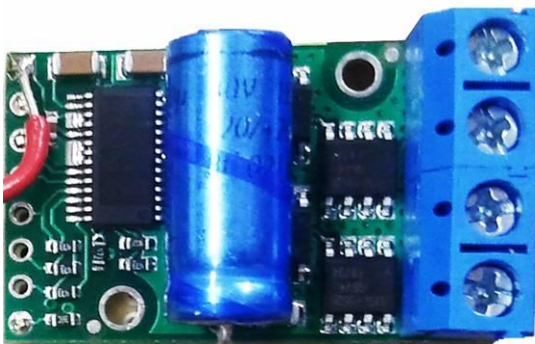


Figure 9 Pololu High Power Motor Driver 18v15

Nominal Maximum Voltage and 15Amp maximum continuous current without a heat sink. This meets our window motors requirement as they operate at 12V and should draw less than 5 Amp at maximum load. These motor drivers also have a digital motor direction input (DIR) and the motor speed is determined by a PWM input pin. Additionally, the regulated 5V DC supply pin would provide the supply voltage for the photo interrupters (maximum $V_{cc} = 7V$), a crucial portion of our design that will be discussed in the following section. The logic connections, especially PWM pins, are designed to tolerate a minimum “high” input signal threshold of **3.5 V**. This was a potential problem as the Digital I/O of the Arduino Fio was **3.3V**. To address this issue, we explored the option of adding a voltage boosting circuit, such as using an Operational Amplifier. However, as we were testing the data on the H-Bridge, we noticed that, despite what was written on the official specification, the “High” versus “Low” cutoff of these H-Bridges were in fact around **2.2V**. This discrepancy resulted in data matching between the Arduino and the motor drivers. As we proceeded with more testing on these drivers, we discovered that their wiring design was not ideal. During one of our testing sessions, two out of the four drivers stopped functioning permanently, while one of them caused a breakdown of an XBee radio as well. After analyzing the testing and operations we had performed, a theory was induced that whenever a connection is removed from the 5V pin, the internal circuit will break down.

Therefore, we decided that we shall not use these 5V pins to avoid similar outcome. Since we had to replace these components, we exceeded the budget limit (each driver costs \$39.95 + shipping). As we tried to test these drivers on our actual window motors with no load, we recorded a draw of less than 5A. The motor was running for a short period and then completely stopped. Since the motor was one of the most expensive equipment in our design, we used another motor driver to test it and obtained the same outcome. Upon further testing, we recognized that these drivers had again broken down. Due to the lack of possible alternatives, we determined to construct a demonstration that would emulate our design concept, with the focus on wireless control and the motor feedback system.

A system was setup so that the receiver Fio would interpret which direction the motor is rotating or if it has stopped either manually or when the arm encounters an obstacle. A chopper is attached to the motor shaft which passes the light paths between two consecutive photo interrupters.

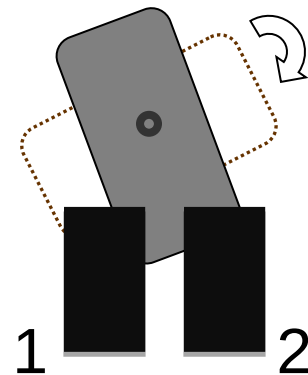
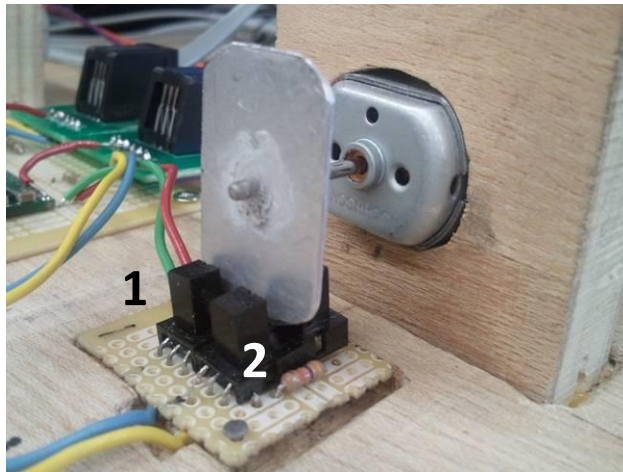


Figure 10 Left: Motor Chopper & sensors. Right: Graphical Illustration

As shown above, when the chopper passes through sensor “2” then sensor “1”, it indicates that the motor is rotating clockwise, and vice versa. PWM signals are generated from Vout of each photo interrupter if the motor is rotating continuously. That is, the output signal would be “LOW” across the width of the chopper and “HIGH” when the light path is not blocked. The Arduino Fio has two digital I/O pin (D2 and D3) that could be configured for interrupt. Combining these features, we could service an interrupt and

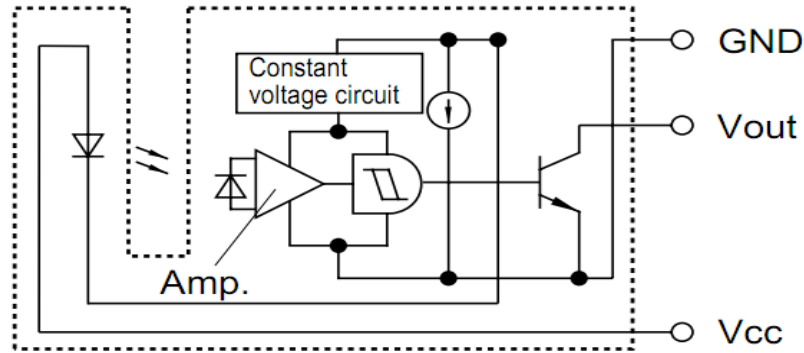


Figure 11 [GP1A73A Schematic](#) ⁵

accurately determine the positioning of the motor arm. The coding portion of the feedback system will be discussed in the software section of this chapter. Figure 4 showed a demonstrative setup of the feedback system, and this will be implemented in the final product when the robotic arm is built. With the help of Mr. Blosser, we constructed the same detection circuit on the window motor (see figure 6). All motors that direct the position of the arm movement (i.e. excluding gripper rotation and open/close motion) will have this feedback system installed.

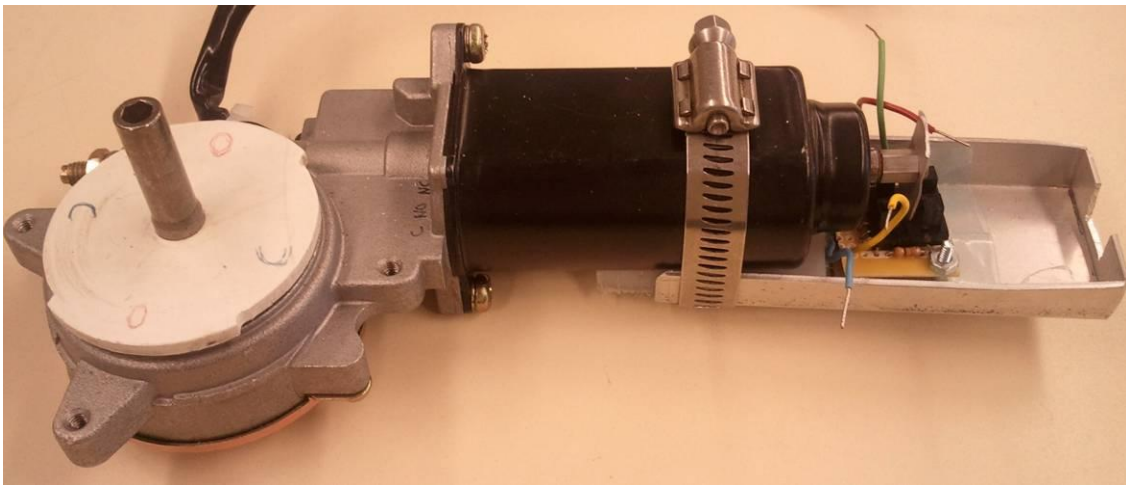


Figure 12 Modified Window Motor with added casing, chopper and

This system serves two crucial functions. First of all, when the system detects a direction using these two photo interrupters, it updates the position value stored in the Arduino program. For example, if the X axis motor moves the arm to the right, then each discrete step of this movement would increment the X-position value, and if it moves left then the value would decrease in discrete steps. On the final product, aside from knowing the exact position of the arm, these position values could be used to detect when the arm has reached its end points. As these end points are reached, the program would ignore a

joystick input of the same direction. This means that the user could not accidentally hit the joystick and push the arm against the limiting ends. The second crucial function of the system feedback is that the program could use a timing poll to detect obstruction. For example, if the arm has not reached any of the end points, then a joystick input should be able to move it. If the feedback system does not detect any changes in the position value within a given polling period, then it concludes that the arm is being stalled by certain obstruction. The program could now ignore any joystick input to avoid unnecessary stall current. On the other hand, since the controller Arduino is inside the controller console, an additional Digital I/O pin could be used to send a warning signal (e.g. lit sign or buzzer) telling the user to discontinue current operation. This also addresses another safety issue – when the arm is handling a load that exceeds its capacity.

The controller Fio will be programmed to interpret various inputs from the joystick controller. The variation is controlled by the potentiometers of the analog joysticks. Since the reference voltage of the Arduino board is set to 3.3V, we use this same voltage which is available on the Arduino board (3V3 pin) as the voltage source for the potentiometers of the joystick. There are two potentiometers on these joysticks (figure 7), one along the X direction and one along the Y direction. When the shaft is pulled to

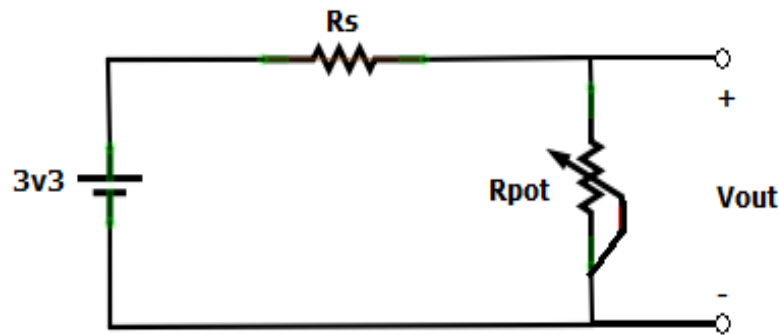


Figure 13 Equivalent Schematic for each joystick direction

the far left of the X direction (or bottom for the Y direction, figure 8), it is equivalent to the potentiometer tuned up in figure 7, and Vout would be 0V (Rs is small and the

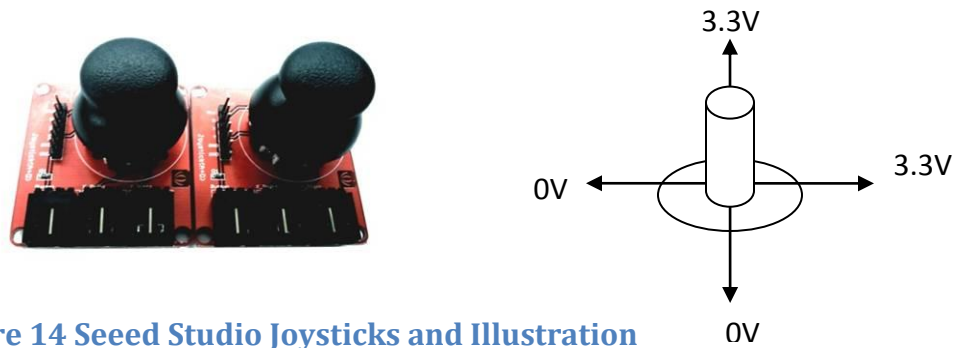


Figure 14 Screenshot Studio Joysticks and Illustration

voltage drop across it is negligible). This voltage output increases linearly as the shaft moves towards the other end (see figure 8, X – Right, Y – Up), and this is equivalent to the potentiometer being at the bottom position in figure 7. This simple design allowed us to feed an easily manipulated signal back to the analog pins on the Fio. There are 8 of these analog pins available that calculate the input value using a **10bit resolution**. Since this value is used to direct a corresponding 8bit PWM output (transmitted to the receiver to control motors) that is 8bit, a simple “divide-by-two” calculation would convert the value correctly. Since our design focuses on wireless control, we constructed a controller console that is user-friendly for people with quadriplegia. The original joystick shafts were too small to handle by people with limited muscular control. Therefore the team replaced them with slightly bigger shafts crafted with soft plastic material (See figure 9).

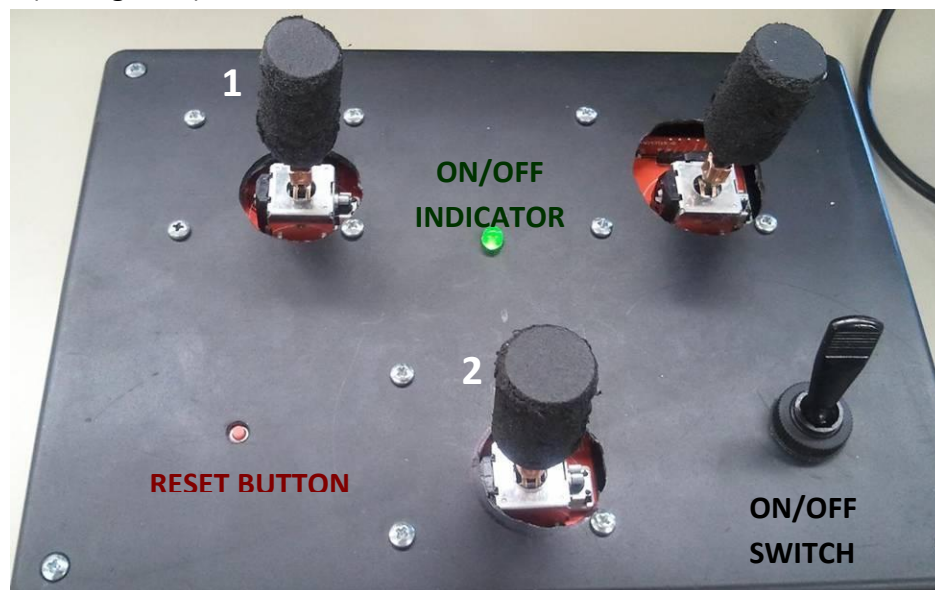


Figure 15 Controller Console (Top View)

The Arduino Fio has ON/OFF pins that could be soldered to external switches. Additionally, the DTR pin on the Fio could reset the board when it is brought low. A

grounded reset button was therefore added in series with the DTR. When the previously mentioned “warning indicator” is on, the user could simply press the reset button to stop all ongoing procedure. This could also be used when unexpected bugs or malfunctioning occur. In future implementation, an improvement to this function could add a “home” feature. This means a reset would return the arm to its allocated default position.

Utilizing what we know about woodwork and our mechanical skills, we created a demonstration system that consists of:

A Control Box with USB power socket, joysticks, buttons, securely enclosed circuit and rigidly positioned Fio Board and battery using non-conductive woodblocks.

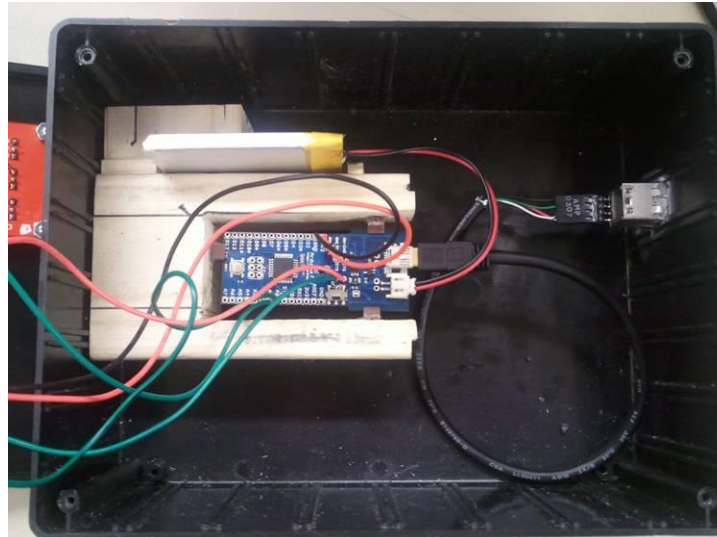


Figure 16 Demonstration System

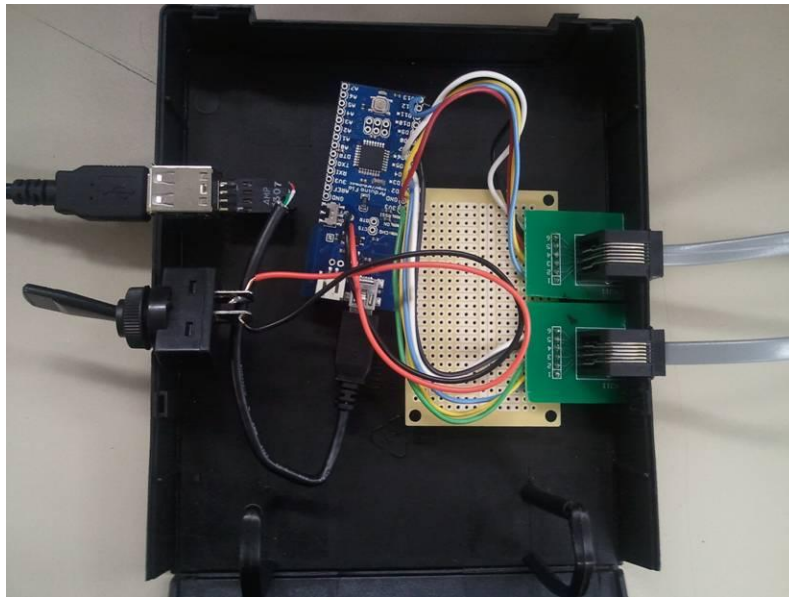


Figure 17 Receiver Box

The receiver box containing two RJ-11 6-Pin connectors, a power switch, USB power connector and the receiver Fio board.

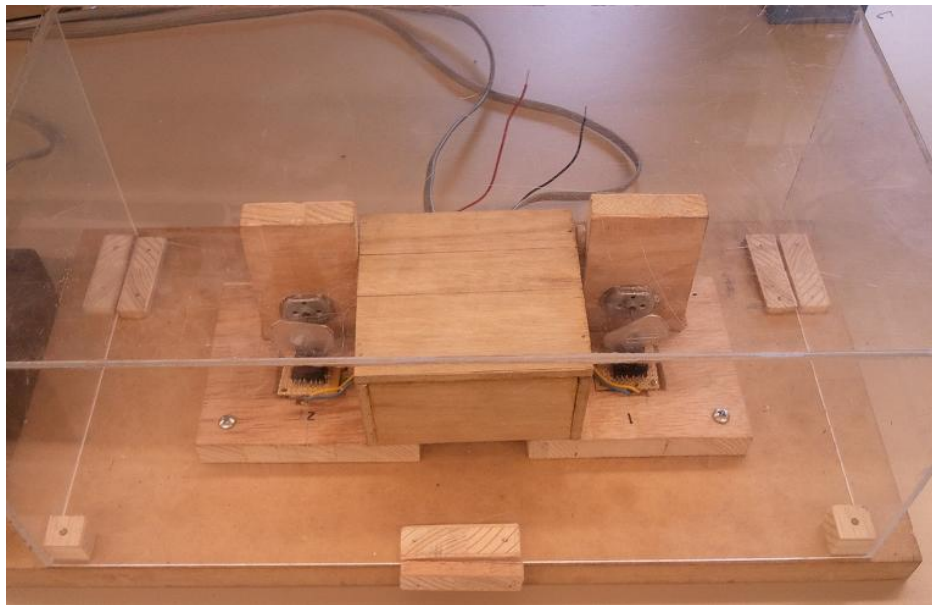


Figure 2 Display Box

A Display Block with enclosed circuitry where motors, choppers and photo interrupters are exposed for demonstration purposes.

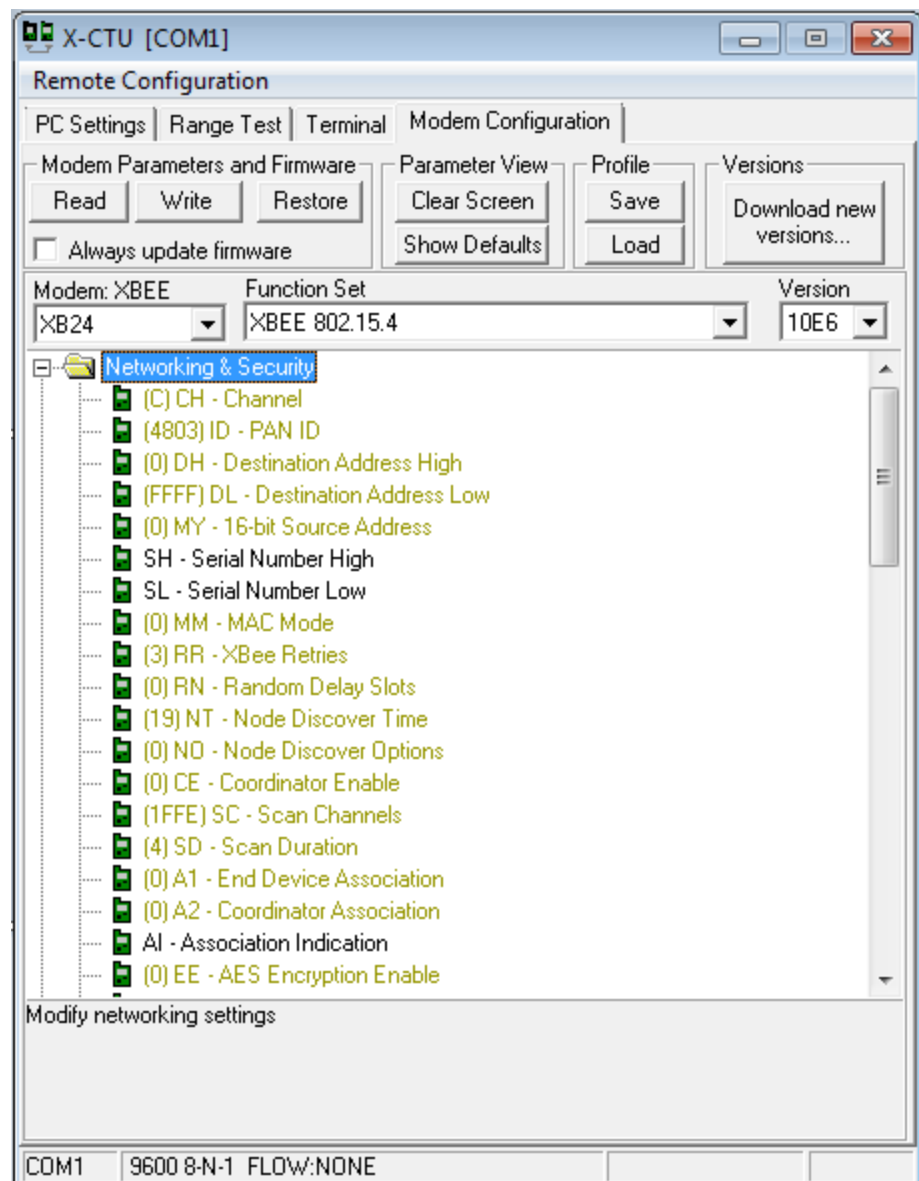
Software Implementation

Various software packages were used in developing the final product by the group. The core of the software development was the Arduino programming language and development environment. Also a configuration tool for XBee modems called X-CTU was used, though it did not involve any software development.

X-CTU

The X-CTU configuration tool is a tool provided by Digi-Key, the original manufacturer of the XBee modems. The tool allows access to all 59 of the settings available on the XBee. The tool is harder to use than the simple configuration software available on the Arduino website but it allows much more freedom when using the modems, something this project required because the sender receiver relationship could not be satisfied using the Arduino

Fio configuration tool. It allows profiles to be saved in text format so when a configuration is found that satisfies the requirements it can be permanently stored. These profiles can be loaded and a new XBee can be setup the same way. This makes replacement of broken units very simple



Arduino Development Environment

The Arduino development environment is a simple open source text editor and programming interface for use with Arduino open source hardware. The programming language used is based on C and contains libraries with functions that accomplish almost all of the I/O and setup. To set up pins on the board for input or output the function `pinMode(<pinNumber>,<mode>)` is used, where `pinNumber` is the label on the pin and `mode` is `INPUT` or `OUTPUT`. Serial communication is setup simply by calling the function `Serial.begin(<baudRate>)` where `baudRate` is the speed of the connection in bits per second, common values being 9600 and 57600. Sending and receiving data over the serial link is a simple call to `Serial.read()` which reads one byte of data and `Serial.print(<data>,<format>)`, where `data` is the variable or immediate value to send and `format` is the type to send, for instance `BYTE` to send raw ascii formatted data, `INTEGER`, `CHAR` to send integers or characters. `Serial.print()` sends all data in the data field consecutively.



Along the top of the editor window are a series of buttons that facilitate error checking, loading, storing and programming the Arduino hardware. The buttons, from left to right are: verify, which compiles the program and checks for errors. Stop, which interrupts whatever the program is doing. New, to create new programs, called sketches. Open, to open a sketch file. Save, to save the current sketch. Upload, to compile the sketch and program the connected Arduino with it. Serial Monitor, which will print all character values that appear on the serial line, and allows the computer to send characters over the same serial line. For the wireless programming a small portion of the memory on the Arduino Fio is reserved as a wireless bootloader that allows the current program to be interrupted by a wireless serial connection. The XBee modem disguises the wireless communication as standard serial so no additional configuration is necessary on either the computer or the Arduino to enable wireless communication.

Arduino Code

Controller

The code running on the controller Arduino board is the simplest code written for this project. It consists of a setup loop that sets analog pins 0 and 1 as inputs and a main loop that polls those pins repeatedly. The input pins have ten bit analog to digital converters on them that return a ten bit unsigned integer. When read, each pin value is

divided by four to shift off the lower two bits making them one byte each so that only one serial send operation has to occur. A character is sent in eight bit ascii format followed by the eight bits of data, the character is interpreted in the receiver to determine how the next byte should be handled.

Receiver

The receiver code is more complex than the controller but is still intuitive and easy to understand. It has a similar setup loop which initialized pins for input and output, two outputs per motor and two inputs per motor for a total of four input and four output pins. In the main loop, when two or more bytes are in the serial read buffer, that means at least one letter data combination is ready, so the program tests the next value in the buffer to see if it is an x or a y. If it is then the byte is read and the next byte is tested to see if it is an x or a y. If the next byte is an x or a y then the data was lost from the last send and the next byte is not read. The loop then restarts and tests again, if an x or y is followed by data that data is used to set the duty cycle of the x or y output. The function `analogWrite(<pinNumber>,<duty>)` is used to set up a pulse width modulated signal with duty cycle <duty> at pin <pinNumber>.

The receiver must take a raw input from the controller and shift it into the proper range of values and set the direction pin to LOW or HIGH based on the position of the joystick. The algorithm used is as follows; x or y input data ranges from 0 to 255. 0 - 120 is considered in the negative direction, while 134 - 255 is considered positive. These limits are the center value(127) plus or minus 7. This data is shifted into the range 0 to 255 for output, for the 0-120 range the number decreases, but we want the PWM to increase. To accomplish this the x or y data is subtracted from the upper limit(120) and then doubled, giving an effective range of 0-240. For the range 134-255 the opposite must happen, as the number increases, the PWM must also, so the lower limit(134) is subtracted from the x or y data, giving an effective range of 0-242. Anything within 7 of the center value(127) is considered 0. The direction value is set to LOW for the negative range and HIGH for the positive range.

Chapter 4: Testing and Functionality

Testing was done in several phases. Each part of the project was tested individually starting with the wireless communication. When that was tested and debugged the pulse width modulation (PWM) output capability of the Arduino boards was tested. That was followed by Analog input and interrupt handling tests, which concluded basic testing of the microcontrollers. The next tests were run on small DC motors using motor drivers from Pololu Robotics using a function generator as input. This was followed by connecting the Arduino receiver outputs to the motor driver and using the wireless communication of analog inputs to spin the motor in both directions. Next, photo interrupters were tested and used to trigger a trivial interrupt service routine on an Arduino, without the motors connected. The photo interrupters, when confirmed functional, were attached to a window motor with a chopper blade on it and tested with a 12V DC power supply that was intended to run the motors in the final design.

Wireless Testing

The first phase tested wireless transmission and reception between the two Arduino Fio microcontrollers. The tests involved three XBee modems, one in each microcontroller socket and one connected via an adapter to a PC. When a Fio modem was configured using the tool available from the Arduino website, the XBee modem would either receive data from any source and broadcast only back to the programmer modem, or would broadcast data to all other devices but not receive from anything. This was because the configuration tool only configures a modem to be a programmer or a Fio, the Fio mode essentially setting it up to work as a remote sensor. A series of tests were run using the tool from Digi-Key, the XBee modem's original manufacturer, called X-CTU. This tool allows all settings to be changed or read off of a preconfigured modem. A modem configured as a programmer and one as a Fio were read and compared to begin the test. X-CTU can generate a text file containing the settings, these files were compared line by line to determine what settings were different and why. In total seven different fields were different between the modems configured as programmer and the one as a Fio. The following table contains the field names and values.

Field	Programmer	Controller	Receiver
DL	0xFFFF	0xFFFF	0
MY	0	1	2
RR	3	3	0
D3	3	3	5
IU	1	1	0
IC	8	8	0
IA	FFFFFFFFFFFFFFFF	FFFFFFFFFFFFFFFF	0xFFFF

DL is the lower thirty two bits of the destination address, 0xFFFF or less if only sixteen bit addressing is being used, with 0xFFFF meaning broadcast to all. MY is the source address of the modem and the address that must be encoded for it to receive any data. RR is the number of retries the modem will attempt before dropping a packet, it is three for the two radios that are sending data, the other radio is not intended to send so it is set to zero. D3 is the configuration for digital I/O pin three of the XBee modem, 2 means analog to digital conversion, 3 means digital input, 4 means digital output 0 and 5 means digital output 1. IU is I/O enable, a 0 means disabled while a 1 means enabled. IC is digital I/O change detect, it is eight bits and each bit corresponds to one of the eight digital I/O pins. IA is the input address, if all eight bytes are set to FF it will not allow any input to change the output, if it is set to 0xFFFF it will allow any input to change the output. IA set to FxXXXX makes wireless programming of the Arduino impossible. The receiver and programmer settings did not have to change, but the controller had to be reconfigured. The final configuration is as follows.

Field	Programmer	Controller	Receiver
DL	0xFFFF	0xFFFF	0
MY	0	1	2
RR	3	3	0
D0	3	3	0
D3	3	5	5
IU	1	0	0
IC	8	8	0
IA	FFFFFFFFFFFFFFFF	0xFFFF	0xFFFF

The programmer sends data over D0 and D3 which is received by both the controller and receiver on D3, the controller uses D0 to broadcast data. The highlighted sections in the table indicate fields that were changed.

Pulse Width Modulation Testing

The receiver Arduino was loaded with simple code that went through a loop that increased the duty cycle from zero percent to one hundred percent over about a second and then did the reverse. The line of code that accomplished that is; `analogWrite(<pinNumber>,<dutyCycle>);` where pinNumber is the output pin, only pins three, five, six, nine, ten, and eleven support pulse width modulation and dutyCycle is an eight bit integer where zero is zero percent duty cycle and 255 is 100 percent. The output was attached to an LED which started off and slowly brightened then dimmed back to off repeatedly. When this test was complete it was run again but increasing and

decreasing the duty cycle at five outputs simultaneously. All five LEDs reacted as expected.

Analog Input Testing

For these tests the controller Arduino was loaded with code that is essentially the same as the final code for the project. The first test tested only a single input which was the output of an analog joystick. The Vdd and Gnd pins on the joystick were connected to the 3v3 and Gnd pins of the Arduino. For the single input the number returned from the ten bit analog to digital converter was reduced to only eight bits by dividing by four and then transmitted over the serial wireless. The receiver was loaded with very simple code that constantly read data from the serial buffer and wrote it as the dutyCycle argument of analogWrite(). To expand the tests a simple serial communication protocol was implemented where a single character is transmitted followed immediately by a single byte of data corresponding to the analog input. The controller was connected to additional joysticks, and the receiver code was updated with an if else statement that checked for at least two bytes in the serial buffer and then read them both determining the letter and then writing the data following it to the correct pin. The receiver was connected to the same set of five LEDs, which got brighter and dimmer with the joystick inputs to the controller Arduino.

Interrupt Testing

To test the interrupt capabilities one of the Arduino boards was configured to be basically a pulse generator and the other was using those pulses as interrupts at two different pins. The first Arduino sent five pulses at one pin with a delay between each one then a longer delay followed by five pulses at a second pin with the same delays. The second Arduino had pins two and three, the two interrupt pins, wired directly to the two output pins of the first Arduino. The interrupt service routine for pin two decremented a global value and then printed it over serial while the routing for pin three incremented the same variable. That variable was initialized to ten and the main loop blinked an LED at five Hz. The interrupt service routines also checked the variable value after changing it, and if it was above ten or below five they turned on an LED, which at no point in the code was turned off, except on initialization. The LED indicated a missed interrupt, because if an interrupt was missed then the increment decrement cycle would become unbalanced and the value would break out of the bounds five to ten. At up to 1.4 kHz the second Arduino ran for twenty minutes and never set the error LED, meaning the board could handle interrupts up to 1.4 kHz with zero percent loss.

First Motor Driver Tests

With the controller and receiver working, motor drivers were tested with small DC motors attached to them. The motor drivers take a pulse width modulated signal for speed and a DC signal for direction, translating that into a percentage of the external source voltage across two output terminals with the direction signal changing the polarity of it. This results in the duty cycle changing the speed and the direction signal changing the direction the motor turns. The PWM pins of two motor drivers were attached to two of the PWM output pins on the Arduino and the direction pins were attached to two of the digital I/O pins on the Arduino. Code was reloaded onto the boards that read the analog joystick values and sent them to the second Arduino which then generated the control signals for the motor drivers. The motor driver V+ and Gnd terminals were connected to an Agilent power supply set to 5V with a current limit of 500mA. The first run failed because the ground of the motor drivers and the Arduino were not common. When that was fixed, the tests worked as expected with both motors spinning with variable speed in both directions.

Photo Interrupter Testing

Testing the photo interrupters involved first wiring them correctly, each requiring a 47k Ω pull up resistor between the Vout pin and Vcc. When the sensor is interrupted, Vout equals Vcc, when it is not Vout is roughly equal to ground. After confirming the voltage outputs with a multimeter, two interrupters were wired next to each other and soldered into a bread board with their resistors. This sensor configuration allows for direction detection by one triggering an interrupt and the service routine reading the value of the other. If the interrupting object is moving from the interrupting sensor's side toward the other then when the interrupt is caught the second sensor will not be interrupted yet and a low voltage will be read. If the object is travelling the opposite direction, the second sensor will already be high when the interrupt is caught. The pair of interrupters was wired into an interrupt and digital I/O pin on the receiver Arduino with the service routine lighting up an LED if interrupted in one direction, and turning it off in the other direction. This worked as expected, even while reading wireless data and sending PWM signals to the two motors.

Power Supply and Window Motor Testing

To test the window motors, a larger supply was needed, so the 12V DC power supply was also tested at the same time. After measuring the outputs of the 12V DC supply with a multimeter to confirm the voltage, a window motor replaced the two small DC motors from the previous tests. For the first test the pair of photo interrupters was mounted to the end of the motor with a chopper blade attached to the drive shaft so that it would interrupt the sensors as the motor spun. The motor was attached directly

to the supply with no driver or Arduino and the power and ground from the sensors was attached to an Agilent power supply set to 5V. With the motor on, two out of phase square waves appeared on the oscilloscope. For the next test the motor driver was put between the source and motor and the sensors' power lines were attached to the motor driver's +5V and Gnd pins while the signal lines were attached to the Arduino input pins. The motor driver was attached to the Arduino outputs, but the motor would not run. Attaching the photo interrupters to the +5V pin of the motor driver had somehow pulled enough current to burn out an IC on the board. A second motor driver was used to determine the cause of the failure and that driver stopped functioning after a multimeter was attached to the +5V pin. Power for the sensors was going to have to be run from the Arduino itself. This configuration was changed, and the sensors continued to operate correctly. The testing then continued with the sensors attached to the Arduino and the motor attached to the driver outputs. The motor ran correctly for a couple of minutes when another driver burnt out. After further testing a fourth driver failed. Two replacements had been ordered after the first round of failures, and those drivers were set aside for lower power uses later. No suitable replacement could be found within the time requirements or remaining budget for the project and the group sponsor confirmed the datasheet for the motor drivers indicated they would work for this application. At this point a new demonstration design had to be made and implemented.

Demonstration Design and Testing

The final demonstration hardware for this project consists of a completed wireless control interface, position and direction sensing, and variable speed and direction control of two motors simultaneously. The controller consists of a large plastic box with three joysticks, one of which is connected to analog inputs, a reset button, power indicator LED and switch and a USB charge port. The receiver has a power switch, USB power supply and two six wire RJ11 terminals for communication with the motors and sensors. The motor assembly has two small DC motors mounted in wood blocks with metal chopper blades on them that each rotate through a pair of photo interrupters, with all control signals and power and ground for the sensors coming from two six wire RJ11 ports on the back. Two additional power terminals are on the back for powering the motor drivers and consequently the motors. The motors are covered by a large clear Plexiglas box so that the motion can be seen without danger from moving parts. The test platform was tested by powering all components and attaching an oscilloscope to the signal lines from each pair of sensors with the motors moving in different directions.

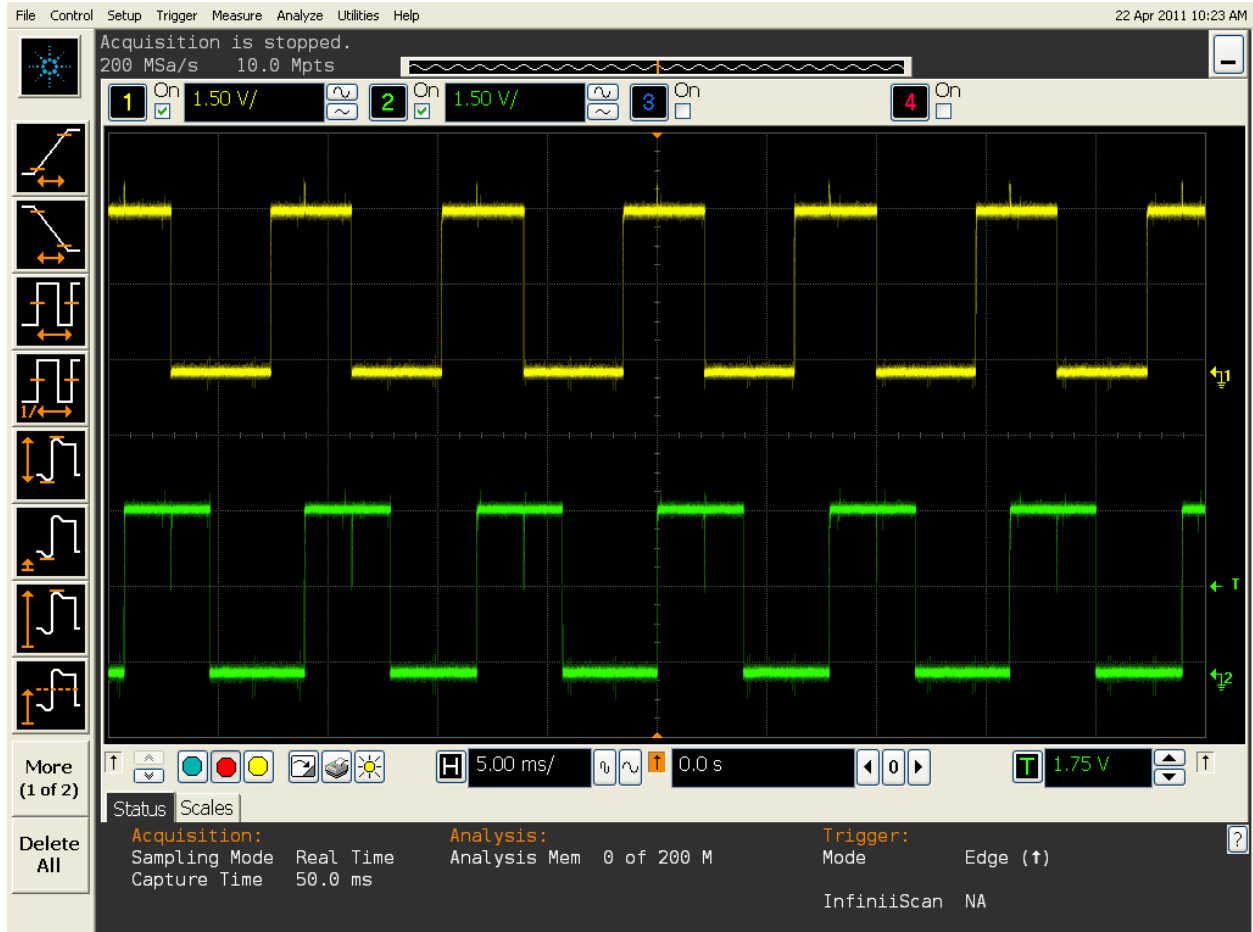


Figure 19 - Motor Feedback: Clockwise

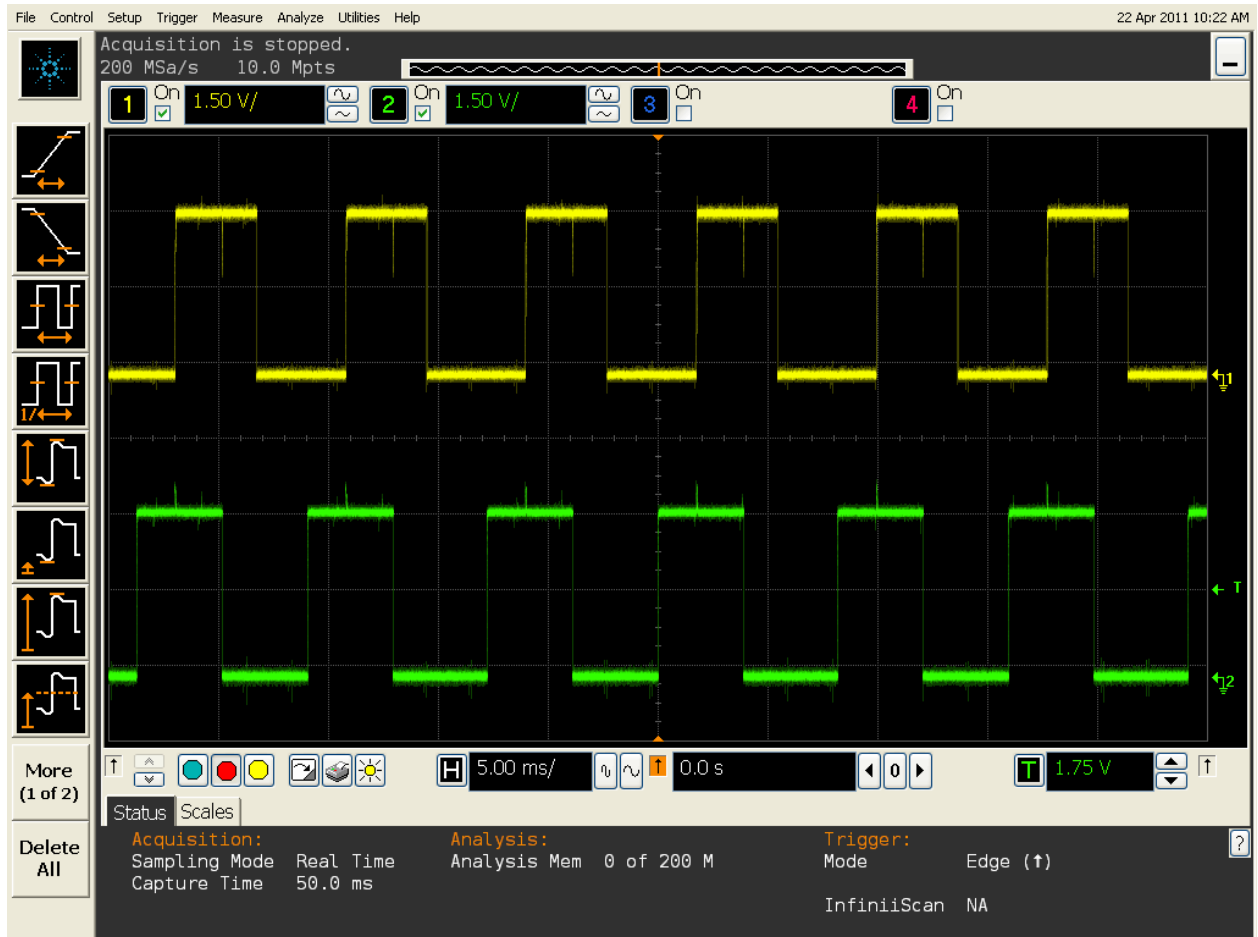


Figure 20 - Motor Feedback: Counter-Clockwise

Chapter 5: Conclusion

Early on in the design process of the robotic arm, our team realized that the lofty goal of building both the mechanical structure and the joystick interface might not be realistic due to the time constraints of the semester and the limited budget. Realizing this, our sponsor representative Professor Blosser along with a mechanical engineer volunteer, offered to build our design of the mechanical structure of the robotic arm. This left our team with the goal of building a wireless joystick interface, achieving motor control with sensor feedback, and acquiring a gripper for the lifting of lighter payloads. In each of those goals, our team experienced successes and failures that both reinforced our own knowledge and sometimes made us rethink our approach. Through research and experimentation our team learned that we need to be adaptable in the designing and building process to be successful.

The first objective our team decided to tackle involved developing a wireless joystick interface. Our team chose to establish wireless communication using two Arduino FIO boards with XBee radio attachments. Our first success occurred when we successfully programmed an Arduino FIO board wirelessly to blink the LED with a chosen frequency. This eventually led to our team establishing wireless communication between each of the boards. For motor control our team needed to produce a pulse width modulated (PWM) signal with varying duty cycle. To accomplish this, our team implemented analog joysticks at the transmitter Arduino FIO board. The transmitter board would read in the voltage from the analog joysticks and wirelessly transmit a signal that would vary the duty cycle of the PWM signal at the receiver board. After many trials, our team was able to successfully build a wireless joystick interface which would produce PWM signals that could be used for motor control.

For control of the motors, our team decided to implement H-Bridge motor drivers that use the PWM signals from the receiver Arduino FIO board to control our 12V DC window motors. The H-bridge motor drivers our team chose to implement were the Pololu 18V 15A motor drivers. Initially using 6V DC motors, our team was able to successfully achieve both speed and direction control. However, after scaling up to the 12V DC window motors, the motor drivers burned out after a few minutes. Our team suspects that although the MOSFETs of the motor drivers were rated for 15A, the control IC could not handle the increased current the 12V DC window motors were drawing. When our team realized that the motor drivers could not handle the window motors, we lacked the time and the funds in our budget to customize our own. In hindsight, our team realized that we could have possibly avoided this failure if we had built our own. This experience taught us that not all commercially available circuits are reliable.

While establishing motor control, our team decided to implement a sensing system called a tachometer to provide feedback to the Arduino FIO receiver board. To do this our team used two IR interrupters in conjunction with a chopper wheel. In this process, the chopper wheel would interrupt the IR interrupters. The signals resulting from the sensing system were two square waves with a phase difference of about 90 degrees. These signals were used to determine the speed from the frequency of the square wave. The direction of the motor is determined from which square wave is leading. Our team successfully customized a sensing system that could be used not only for sensing speed and direction, but also the position of the robotic arm.

For the next design team that will take over this project, our team recommends that they should build the H-bridge Motor drivers instead of purchasing them. Our team learned to our detriment that not all commercially available motor drivers are reliable. In research conducted by our team, other motor drivers that claim to handle the current of the window motors cost over \$90 for one motor driver. To save money and prevent a potential catastrophic failure to the project, a customized H-bridge motor driver should be built instead of bought. Another suggestion our team would make to the next, would be to program the robotic arm to have Cartesian movements to make the robotic arm more user friendly. The Arduino board would need to be programmed to calculate position of the robotic arm and run at most two motors at any one moment. Our team did not have the chance to write the code due to the time constraints and the fact that the robotic arm being built simultaneously as the rest of our project.

The first six weeks of the semester was spent designing the mechanical structure of the robotic arm and researching possible parts that our team could implement in the project design. Our team then spent the three weeks after that developing our wireless joystick interface, while also researching and ordering the motor drivers and sensing system needed for the project. During that time we also constructed the gripper and determine the signals needed to control the gripper. The following three weeks were used to try to accomplish motor control using H-bridge motor controller and developing the sensor system. After the failure of the motor drivers, in the last 2 weeks our team decided to develop a demo for design day featuring the sensing feedback system with the remaining motor drivers.

Throughout the semester, our team learned to work with each other and persevere through the difficulties we faced in designing and building this project. Though our team did not accomplish every goal we set for ourselves this semester, we each feel that we did our best to complete as many goals as we could. The experience gained from

working on this project and working with each other has made each of us a better engineer.

Final Cost

Part Name	Unit Price	Quantity	Item Total
Arduino FIO	\$24.95	2	\$49.90
XBee Radio	\$22.95	4	\$91.80
Polymer Lithium Ion Battery (2000 mAh)	\$16.95	1	\$16.95
Wall Charger	\$3.95	1	\$3.95
Seeed Studio Analog Joystick	\$7.90	3	\$23.70
Window Motor Gear Motor	\$19.99	1	\$19.99
Pololu High-Power Motor Driver 18v15	\$39.95	4	\$159.80
Replacement Pololu High-Power Motor Driver 18v15	\$19.97	2	\$39.95
XBee Explorer USB	\$24.95	1	\$24.95
Power Supply 12V 29A	\$47.00	1	\$47.00
DC-to-DC converter	\$6.20	2	\$12.40
Gripper(no servos)	\$17.89	1	\$17.89
Standard Servo	\$17.99	1	\$17.99
Male USB to Male Mini USB- 3ft	\$9.01	2	\$18.02
Male USB to Male Mini USB- 6ft	\$23.99	1	\$23.99
Power Switch	\$2.99	2	\$5.98
Project Box 8x6x3	\$7.00	1	\$7.00
Photo Interrupter	\$1.10	10	\$11.00
Total			\$592.44

Appendix 1 – Technical Roles



Thomas Manner

Tom did the configuration of the wireless communication radios for the XBee modems used in the controller interface. This involved a series of tests in the firmware configuration of each radio and in the software running on each Arduino Fio. He also developed all of the software for the project, including the controller and receiver code, as well as the computer application used to demonstrate that motor feedback is functioning correctly. Tom was very involved in the early design work, both the conceptual designs of the arm and the concrete design of the control system. He often acted as a translator between the team sponsor, Mr. Blosser, and the other members of the team when a mechanical concept was discussed. He was instrumental in fabricating the test platform because he has more experience with workshops and power tools than most people, having built several projects from wood using a band saw and drill press.

Tom did much of the soldering once circuits had been developed and tested in protoboards including making two of the photo interrupter circuits and the circuit that routes the motor control signals in the final demo. The photo interrupters are from Digi-Key and were \$1.10 apiece with a simple three pin interface, his research led to the purchase of them as the sensors for all the motors.



Daniel Phan

The main technical role for Dan in this project involved power management of both the 12Vdc window motors and sensing systems used in the robotic arm. Considerations towards the voltage and current drawn by the window motors, led him to research for a power supply to fulfill those needs. He had to consider that the robotic arm would require up to two motors running in parallel. This led him to conduct experiments in the lab to test how much current the window motors would draw. After conducting extensive research, the team purchased a power supply that could provide 12Vdc and up to 29A of current. The power supply would convert

120Vac from a wall outlet to a 12Vdc source. This powersupply was able to power the DC power supply.

To power the sensing system when installed in the robotic arm, Dan had to consider how to step down the 12Vdc source to 3.7Vdc. To accomplish this task, he research several DC-to-DC converters that could possibly be used in this scenario. He ended up choosing and buying a DC-to-DC converter that can handle up to 14Vdc input and output a voltage range between 600mVdc to 6Vdc. After some testing, he was able to use a 470 ohm resister as a trim resistor to output 3.7Vdc. The converter is able to draw up to 3A of current which is more than the IR interrupter would draw. The research and implementation of this converter provides a voltage supply to power the IR interrupter of the sensing system.

Other areas Dan was involved in included the selection of a gripper. After research conducted by the team, a biped robotic gripper was selected. Dan purchased the gripper kit and constructed the entirity of the biped gripper and installed the servo motor. After the gripper was tested, it was discovered that the servo motor would draw 20mA when idle and over an amp when a person tried to open the gripper. Another area he was involved in was the initial testing of the H-bridge motor drivers. The team researched how this type of motor driver works and he helped to initially test them.



Ali Alsatarwah

Ali was involved in several technical roles throughout the project. He was responsible for most of the research and reporting information about products and devices that are most suitable for the project. It started with internet research on the linear actuators to choose the best design to achieve the purpose of the project within the limit of the budgets. Most importantly, he was the responsible for researching and finding one of the main parts in order to complete our project, which was the H-bridges motor driver. He found H-bridges in the market that have the features required for our project and are affordable. Ali's main role in the project was testing and the demo. He and

Dan worked on testing most of electrical components of our projects in order to determine the limitation and capability of each component. They were a time savers for the team. Ali, while testing the gripper that we intended to use in our project realize that it draw large amount of current comparing with its specifications. As a result of that he found a mechanism to lower the power consumed by the gripper and protect the users from any high current leakage. In addition, Ali and Tom from testing the H-bridge motor drivers discovered two issues with the motor driver that play a role in determining the progress of the project. First issue was that they found that these motor drivers have a pin that, when connected, can cause damage to the entire motor driver, which was not mentioned in the device's date sheet. The second issue was that Ali realized from testing these motor drivers, they are not capable of driving and controlling high power motors, which conflicted with the data sheet. By this discovery, the team realized that it is unrealistic to complete the project as expected. As a result of that, Ali figured out a representation for the project and tried to visualize a model or a dome for the project using the available parts

Another Ali's technical role was also building prototypes for circuits and logic gates that we used for testing the project on small scale. From his prototypes the team was able to make the project subsystem interface on low power scale and spot the areas that needed some improvement. Ali also worked on testing the wireless range of the controllers. In addition he contributed on troubleshooting the subsystems of robotic units such as the controlling system, the sensory systems, and the electrical power system. Whenever there was an error or faulty system, the team referred to Ali and Tom to figure out what the source of the issue was and make the system work again. As the end of the semester approached, Ali worked with the team to fully integrate all the subsystems that he helped to develop into one system that represents the project requirements.



Ka Kei Yeung

Ka Kei Yeung researched on various possible components while carrying the principle of lowering the cost of production. He first found a cheap commercial product called RoboStir that would immediately solve one of the objectives. However, this objective was found to be unnecessary after a meeting with Doug (the primary user). He later discovered a model of Analog Joysticks that make up the major portion of the controller design. Ka Kei participated in many of the testing procedure. In particular, as he was verifying crucial specification of the H-Bridge motor drivers, he discovered the discrepancy of the Logic input threshold

between the model and its public data. When failure and breakdown occurred on these H-Bridges, he researched on possible ways to restore them. He tested the chip for “fault flags” and “reset” and concluded that these H-Bridges were not wired ideally. He also discovered the probable model of the IC chip whose model was intentionally undisclosed by the manufacturer Pololu.com. In order to lower costs (as this was one of the major challenges), Ka Kei gathered and recycled used parts from his old printer that could benefit the design. This includes motors, gears, tracks and other materials.

Some of the components used have vague or unavailable specification. Ka Kei would research and contact manufacturer in order to find as much data as possible. Additionally, he disassembled and re-assembled the window motors in order to study its mechanics for further modification (installment of the photo interrupters).

Supervised by Mr Blosser, Ka Kei learned several mechanical as the project would require these skills for assembly. He used several software to sketch both 2D and 3D conceptual designs; as well as schematic of the final demonstration circuit. Since some of our components have limited data and specification, he would create his own schematic library to model the total design.

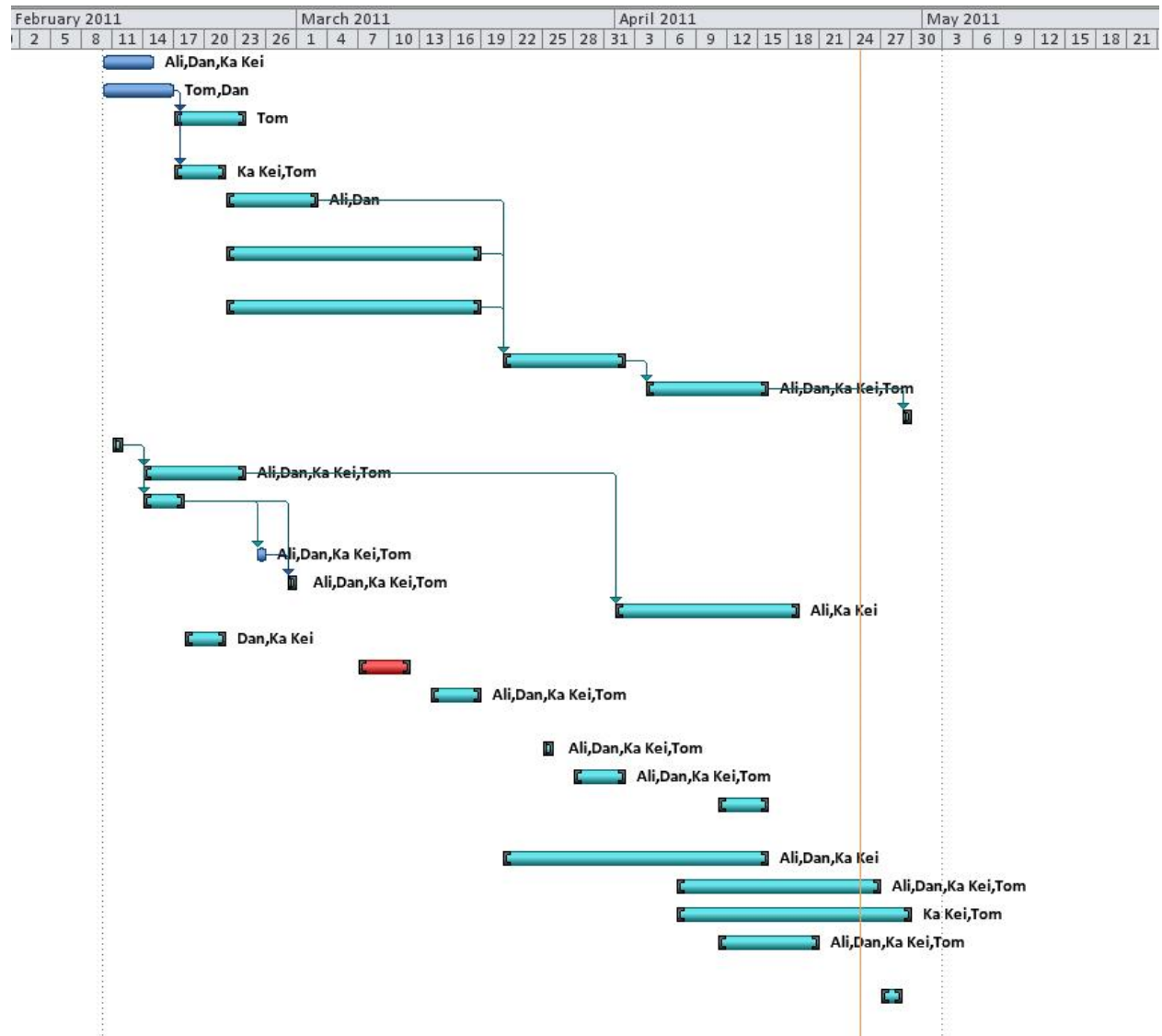
On the final stage of the Design, he participated in much of the mechanical work to assemble the demonstration system. This included the controller box, the receiver box and the display box. He applied his previously learned woodwork and mechanical skills to enhance the robustness of the final display.

Appendix 2 – Gantt Chart

Tasks and Task Assignments

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Gripper Structure Design	3 days	Thu 2/10/11	Mon 2/14/11		Ali,Dan,Ka Kei
2		Arduino and Controller Parts Order	5 days?	Thu 2/10/11	Wed 2/16/11		Tom,Dan
3		Construction, Assembly & Testing (Arduino Parts)	5 days	Thu 2/17/11	Wed 2/23/11	2	Tom
4		Construction, Assembly & Testing (Controller)	3 days	Thu 2/17/11	Mon 2/21/11	2	Ka Kei,Tom
5		Construction, Assembly & Testing (Actuators, Motors, Grippers)	7 days	Tue 2/22/11	Wed 3/2/11		Ali,Dan
6		Construction, Assembly & Testing (X Actuator Extension by Professor Blosser)	19 days	Tue 2/22/11	Fri 3/18/11		
7		Construction, Assembly & Testing (Rotational Arm by Professor Blosser)	19 days	Tue 2/22/11	Fri 3/18/11		
8		Construction & Assembly (Complete Design)	2 wks	Mon 3/21/11	Fri 4/1/11	5,6,7	
9		Create and Test Demo for Design day	2 wks	Mon 4/4/11	Fri 4/15/11	8	Ali,Dan,Ka Kei,Tom
10		Design Day	1 day	Fri 4/29/11	Fri 4/29/11	9	
11		Pre-Proposal Graded	1 day?	Fri 2/11/11	Fri 2/11/11		
12		Final Proposal Writing (Due Feb 25)	8 days	Mon 2/14/11	Wed 2/23/11	11	Ali,Dan,Ka Kei,Tom
13		Create Oral Presentation Slides (Based On Proposal)	4 days	Mon 2/14/11	Thu 2/17/11	11	
14		Oral Presentation Practice	1 day	Fri 2/25/11	Fri 2/25/11	13	Ali,Dan,Ka Kei,Tom
15		Oral Presentation Day	1 day	Mon 2/28/11	Mon 2/28/11	13,14	Ali,Dan,Ka Kei,Tom
16		Poster Design and Submission	12 days	Fri 4/1/11	Mon 4/18/11	12	Ali,Ka Kei
17		Writing FAST Diagram (Due Week 7)	2 days	Fri 2/18/11	Mon 2/21/11		Dan,Ka Kei
18		SPRING BREAK	5 days	Mon 3/7/11	Fri 3/11/11		
19		Progress Report & Photocopies of Engineering Notebook (Due Friday)	5 days	Mon 3/14/11	Fri 3/18/11		Ali,Dan,Ka Kei,Tom
20		Progress Demonstration to Dr. Dong	1 day	Fri 3/25/11	Fri 3/25/11		Ali,Dan,Ka Kei,Tom
21		Application Notes (Due Apr 1)	5 days	Mon 3/28/11	Fri 4/1/11		Ali,Dan,Ka Kei,Tom
22		2nd Progress Report & Demonstration (Due Apr 15) Apr 15)	5 days	Mon 4/11/11	Fri 4/15/11		
23		Design Issues Paper	20 days	Mon 3/21/11	Fri 4/15/11		Ali,Dan,Ka Kei
24		Final Report (Due Apr 26)	14 days	Thu 4/7/11	Tue 4/26/11		Ali,Dan,Ka Kei,Tom
25		Final CD (Due Apr 29)	17 days	Thu 4/7/11	Fri 4/29/11		Ka Kei,Tom
26		Professional Self Assessment Papers (Due Apr 20)	8 days	Mon 4/11/11	Wed 4/20/11		Ali,Dan,Ka Kei,Tom
27		Peer Evaluation (Due May 2nd)	2 days	Wed 4/27/11	Thu 4/28/11		

Project Timeline



Appendix 3 – Controller Arduino Code

```
/*
 * Name:      DemoController.pde
 *
 *
 * Description: Controller code for Design Day demo
 * Created:   April 22, 2011
 * Author:    Tom Manner
 */

/*
 * Function:   setup()
 * Returns:    void
 * Description: Initializes all I/O, runs one time
 */
void setup()
{
  Serial.begin(57600);
  // Set up analog inputs 0 and 1
  pinMode(0,INPUT);
  pinMode(1,INPUT);
}

/*
 * Function:   loop()
 * Returns:    void
 * Description: Runs repeatedly while the board is active, after setup()
 *              runs once
 */
void loop()
{ // Read the analog input pins and shift off four bits
  // of the 10 bits returned
  int x = analogRead(0)/4;
  int y = analogRead(1)/4;

  // Send the character 'x' in ASCII format
  // followed by the x data
  Serial.print('x',BYTE);
  Serial.print(x,BYTE);
  delay(25);

  // Send the character 'y' in ASCII format
```

```
// followed by the y data  
Serial.print('y',BYTE);  
Serial.print(y,BYTE);  
delay(25);  
}
```

Appendix 4 – Receiver Arduino Code

```
/*
 * Name:      DemoReciever.pde
 *
 *
 * Description: Reciever code for Design Day demo
 * Created:   April 22, 2011
 * Author:    Tom Manner
 */

/*
 * Function:   setup()
 * Returns:    void
 * Description: Initializes all I/O, runs one time
 */
void setup()
{
    // Begins serial transmission at a rate of 57600 bits per second
    Serial.begin(57600);

    //
    // Set up x direction sensing I/O
    // interrupt 0(pin2) triggered by x
    // direction sensed by 4
    //
    attachInterrupt(0,xUpdate,FALLING);
    pinMode(4,INPUT);

    //
    // Set up y direction sensing I/O
    // interrupt 1(pin3) triggered by y
    // direction sensed by 13
    //
    attachInterrupt(1,yUpdate,FALLING);
    pinMode(13,INPUT);

    //
    // Set up x direction outputs, 5 is PWM 7 is direction
    //
    pinMode(5,OUTPUT);
    pinMode(7,OUTPUT);
}
```

```

//
// Set up y direction outputs, 6 is PWM 8 is direction
//
pinMode(6,OUTPUT);
pinMode(8,OUTPUT);
}

// X and Y direction variables. volatile means it must reload the
// variable from memory every time it accesses it, so the interrupt
// won't cause incorrect values
volatile int xDir = LOW;//Direction sense
volatile int xPos = 500;//Position, updated in ISR
int xDirOut = LOW;    //Direction indicator(attached to LED)
int xPWM = 0;        //Speed control(0-255)
byte xIn = 0;        //8-bit serial read data

volatile int yDir = LOW;//Direction sense
volatile int yPos = 500;//Position, updated in ISR
int yDirOut = LOW;    //Direction indicator(attached to LED)
int yPWM = 0;        //Speed control(0-255)
byte yIn = 0;        //8-bit serial read data

/*
 * Function:   loop()
 * Returns:    void
 * Description: Runs repeatedly while the board is active, after setup()
 *              runs once
 */
void loop()
{
    // Check for at least two things in the serial buffer
    // because the protocol sends a character followed by a value
    if(Serial.available() >=2)
    {
        // Check the value without consuming it
        if(Serial.peek() == 'x')
        {
            // Consume it if it is the char 'x'
            Serial.read();
            // Make sure the next thing in the buffer is data, and not indicating
            // a different motor
            if(Serial.peek() != 'x' ||
               Serial.peek() != 'y' )
            {
                // Read the data
                xIn = Serial.read();
            }
        }
    }
}

```

```

/*****
x input data ranges from 0 to 255. 0 - 120 is considered in the
negative direction, while 134 - 255 is considered positive.
These limits are the center value(127) plus or minus 7
This data is shifted into the range 0 to 255 for output, for the 0-120
range the number decreases, but we want the PWM to increase. To
accomplish this the x data is subtracted from the upper limit(120)
and then doubled, giving an effective range of 0-240. For the range
134-255 the opposite must happen, as the number increases, the PWM
must also, so the lower limit(134) is subtracted from the x data,
giving an effective range of 0-242. Anything within 7 of the center
value(127) is considered 0. The direction value is set to LOW for
the negative range and HIGH for the positive range.
*****/
if(xIn <= 120)
{
  xPWM = (120-xIn)*2;
  xDirOut = LOW;
}
else if(xIn >= 134)
{
  xPWM = (xIn-134)*2;
  xDirOut = HIGH;
}
else
{
  xPWM = 0;
}
// Values are output, one as a digital and one as a PWM signal
digitalWrite(7,xDirOut);
analogWrite(5,xPWM);
}
}
else if(Serial.peek() == 'y')
{
  Serial.read();
  if(Serial.peek() != 'x' ||
    Serial.peek() != 'y' )
  {
    yIn = Serial.read();

```

```

/*****
y input data ranges from 0 to 255. 0 - 120 is considered in the
negative direction, while 134 - 255 is considered positive.
These limits are the center value(127) plus or minus 7
This data is shifted into the range 0 to 255 for output, for the 0-120
range the number decreases, but we want the PWM to increase. To
accomplish this the x data is subtracted from the upper limit(120)
and then doubled, giving an effective range of 0-240. For the range
134-255 the opposite must happen, as the number increases, the PWM
must also, so the lower limit(134) is subtracted from the x data,
giving an effective range of 0-242. Anything within 7 of the center
value(127) is considered 0. The direction value is set to LOW for
the negative range and HIGH for the positive range.
*****/
if(yIn <= 120)
{
    yPWM = (120-yIn)*2;
    yDirOut = LOW;
}
else if(yIn >= 134)
{
    yPWM = (yIn-134)*2;
    yDirOut = HIGH;
}
else
{
    yPWM = 0;
}
// Values are output, one as a digital and one as a PWM signal
digitalWrite(8,yDirOut);
analogWrite(6,yPWM);
}
}
}

/*
* Function:  xUpdate()
* Returns:   void
* Description: The x direction interrupt service routine. It updates the
*              x position and direction variables
*/
void xUpdate()
{ // Read the x direction sensor

```

```

    xDir = digitalRead(4);
    // Write the direction to the indicator LED pin
    // digitalWrite(12,xDir);
    // Update the x Position according to the direction
    if(xDir == HIGH)
    {
        xPos++;
    }
    else
    {
        xPos--;
    }
}

/*
 * Function:  yUpdate()
 * Returns:   void
 * Description: The y direction interrupt service routine. It updates the
 *              y position and direction variables
 */
void yUpdate()
{ // Read the y direction sensor
  yDir = digitalRead(13);
  // Write the direction to the indicator LED pin
  // digitalWrite(13,yDir);
  // Update the y Position according to the direction
  if(yDir == HIGH)
  {
      yPos++;
  }
  else
  {
      yPos--;
  }
}

```

Appendix 5 – XBee Modem Configuration Files

Modem 0000: Programmer

xb24_15_4_10e6.mxi

FE

0

231

10E6

0

[A]CH=C

[A]ID=4803

[A]DH=0

[A]DL=FFFF

[A]MY=0

[A]MM=0

[A]RR=3

[A]RN=0

[A]NT=19

[A]NO=0

[A]CE=0

[A]SC=1FFE

[A]SD=4

[A]A1=0

[A]A2=0

[A]EE=0

[A]NI=

[A]PL=4

[A]CA=2C

[A]SM=0

[A]ST=1388

[A]SP=0

[A]DP=3E8

[A]SO=0

[A]BD=6

[A]NB=0

[A]RO=10

[A]AP=0

[A]PR=FF

[A]D8=0

[A]D7=1

[A]D6=0

[A]D5=1

[A]D4=0

[A]D3=3

[A]D2=0

[A]D1=0

[A]D0=3

[A]IU=1

[A]IT=1

[A]IC=8

[A]IR=0

[A]IA=FFFFFFFFFFFFFFF

[A]T0=FF

[A]T1=FF

[A]T2=FF

[A]T3=FF

[A]T4=FF

[A]T5=FF

[A]T6=FF

[A]T7=FF

[A]P0=1

[A]P1=0

[A]PT=FF

[A]RP=28

[A]DD=10000

[A]CT=64

[A]GT=3E8

[A]CC=2B

Modem 0001: Controller

xb24_15_4_10e6.mxi

FE

0

231

10E6

0

[A]CH=C

[A]ID=4803

[A]DH=0

[A]DL=FFFF

[A]MY=1

[A]MM=0

[A]RR=3

[A]RN=0

[A]NT=19

[A]NO=0

[A]CE=0

[A]SC=1FFE

[A]SD=4

[A]A1=0

[A]A2=0

[A]EE=0

[A]NI=

[A]PL=4

[A]CA=2C

[A]SM=0

[A]ST=1388

[A]SP=0

[A]DP=3E8

[A]SO=0

[A]BD=6

[A]NB=0

[A]RO=10

[A]AP=0

[A]PR=FF

[A]D8=0

[A]D7=1

[A]D6=0

[A]D5=1

[A]D4=0

[A]D3=5

[A]D2=0

[A]D1=0

[A]D0=3

[A]IU=0

[A]IT=1

[A]IC=8

[A]IR=0

[A]IA=FFFF

[A]T0=FF

[A]T1=FF

[A]T2=FF

[A]T3=FF

[A]T4=FF

[A]T5=FF

[A]T6=FF

[A]T7=FF

[A]P0=1

[A]P1=0

[A]PT=FF

[A]RP=28

[A]DD=10000

[A]CT=64

[A]GT=3E8

[A]CC=2B

Modem 0002: Receiver

xb24_15_4_10e6.mxi

FE

0

231

10E6

0

[A]CH=C

[A]ID=4803

[A]DH=0

[A]DL=0

[A]MY=2

[A]MM=0

[A]RR=0

[A]RN=0

[A]NT=19

[A]NO=0

[A]CE=0

[A]SC=1FFE

[A]SD=4

[A]A1=0

[A]A2=0

[A]EE=0

[A]NI=

[A]PL=4

[A]CA=2C

[A]SM=0

[A]ST=1388

[A]SP=0

[A]DP=3E8

[A]SO=0

[A]BD=6

[A]NB=0

[A]RO=10

[A]AP=0

[A]PR=FF

[A]D8=0

[A]D7=1

[A]D6=0

[A]D5=1

[A]D4=0

[A]D3=5

[A]D2=0

[A]D1=0

[A]D0=0

[A]IU=0

[A]IT=1

[A]IC=0

[A]IR=0

[A]IA=FFFF

[A]T0=FF

[A]T1=FF

[A]T2=FF

[A]T3=FF

[A]T4=FF

[A]T5=FF

[A]T6=FF

[A]T7=FF

[A]P0=1

[A]P1=0

[A]PT=FF

[A]RP=28

[A]DD=10000

[A]CT=64

[A]GT=3E8

[A]CC=2B

Appendix 6 – Photo Interrupter Data Sheet

SHARP

GP1A73A/GP1A73A1

GP1A73A/GP1A73A1

Compact OPIC Photointerrupter with Connector

■ Features

1. Compact type
2. TTL compatible owing to OPIC output
3. Snap-in mounting type
4. 3 kinds of mounting plate thickness
(Applicable plate thickness : 1.0, 1.2 and 1.6 mm)

■ Applications

1. Copiers
2. Laser beam printers
3. Facsimiles

* "OPIC" (Optical IC) is a trademark of the SHARP Corporation.
An OPIC consists of a light-detecting element and signal-processing circuit integrated onto a single chip.

■ Absolute Maximum Ratings (Ta=25°C)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	- 0.5 to + 7	V
*1 Output voltage	V_{out}	- 0.5 to + 7	V
*2 Low level output current	I_{OL}	8	mA
*3 Operating temperature	T_{op}	- 20 to + 75	°C
*3 Storage temperature	T_{stg}	- 30 to + 85	°C

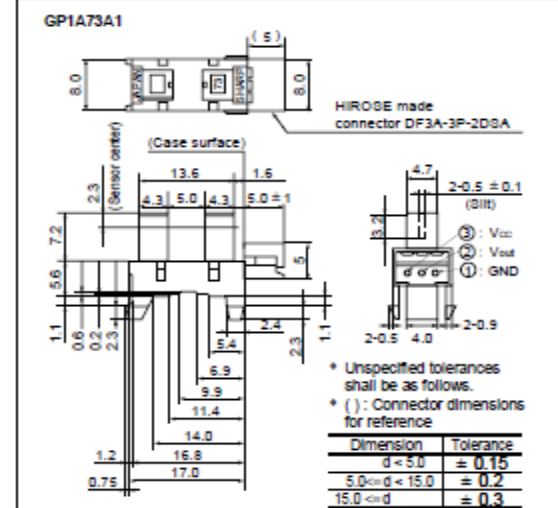
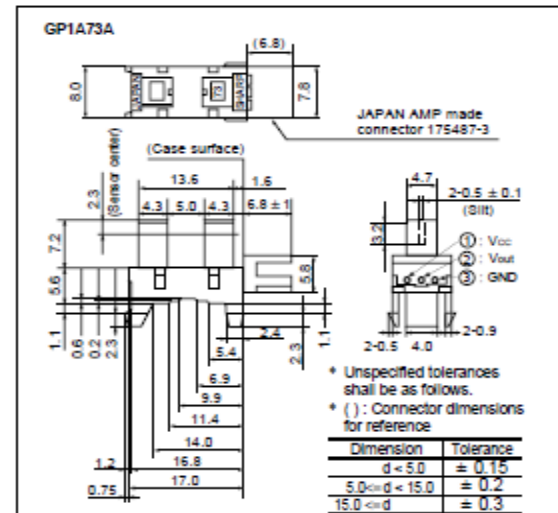
*1 Output transistor collector-emitter voltage

*2 Output transistor collector current

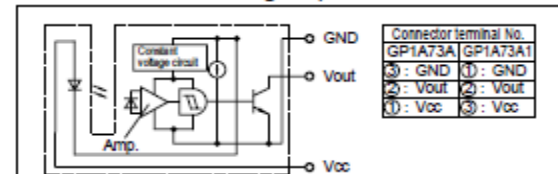
*3 The connector should be plugged in/out at normal temperature.

■ Outline Dimensions

(Unit : mm)



■ Internal Connection Diagram (Both GP1A73A/GP1A73A1)



* GP1A73A and GP1A73A1 are different in the terminal marking number of connectors from each other due to use of connectors of different manufacturers.

In the absence of confirmation by device specification sheets, SHARP takes no responsibility for any defects that occur in equipment using any of SHARP's devices, shown in catalogs, data books, etc. Contact SHARP in order to obtain the latest version of the device specification sheets before using any SHARP's device.*

■ Electro-optical Characteristics

(V_{CC} = 5V, T_a = 25 °C unless otherwise specified)

Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Operating supply voltage		V _{CC}	-	4.5	-	5.5	V
Current consumption		I _{OCL}	Light beam uninterrupted	-	-	16.5	mA
Low level output voltage		V _{OL}	Light beam uninterrupted, I _{OL} = 4mA	-	-	0.35	V
Current consumption		I _{OCH}	Light beam interrupted	-	-	16.5	mA
High level output voltage		V _{OH}	Light beam interrupted, R _L = 47k Ω	V _{CC} × 0.9	-	-	V
Response characteristics	MIN. interruption time	t _H	R _L = 4.7k Ω	166	-	-	μs
	MIN. sensing time	t _L		166	-	-	μs

Fig. 1 Output Current vs. Ambient Temperature

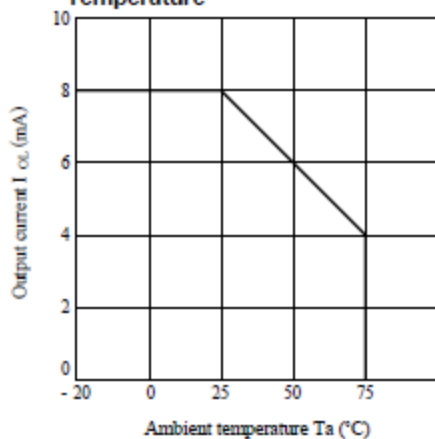


Fig. 2 Low Level Output Voltage vs. Low Level Output Current

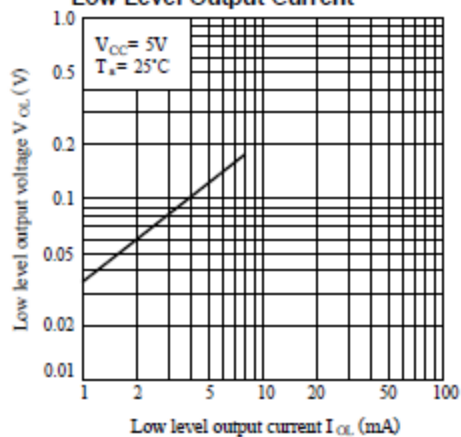


Fig. 3 Low Level Output Voltage vs. Ambient Temperature

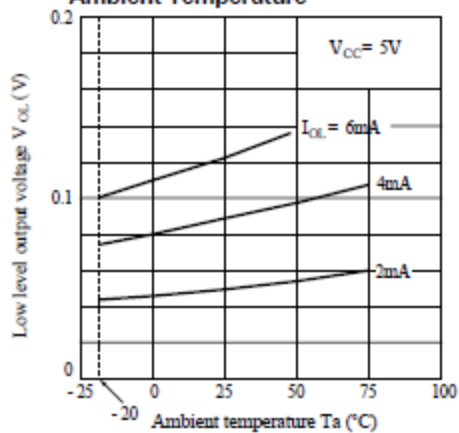


Fig. 4 Supply Current vs. Supply Voltage

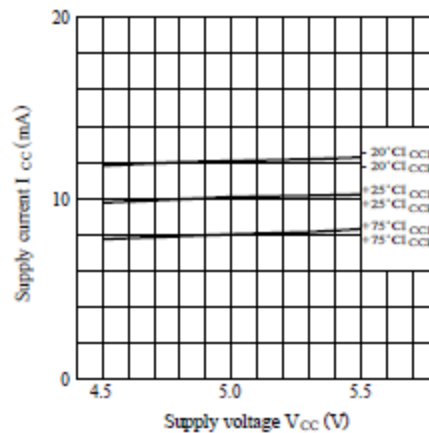


Fig. 5 Detecting Position Characteristics (1)

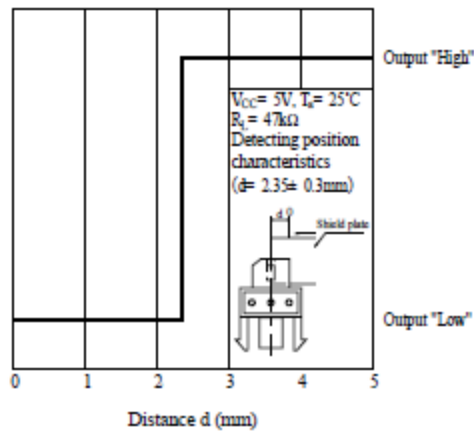
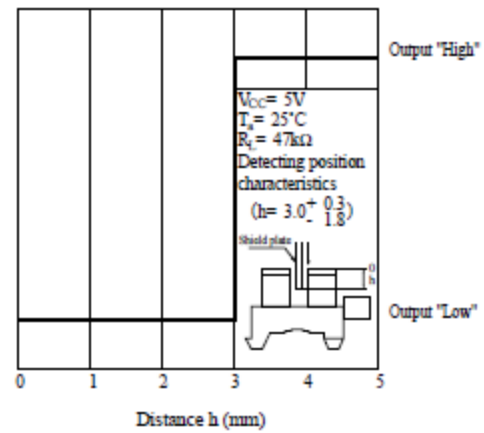
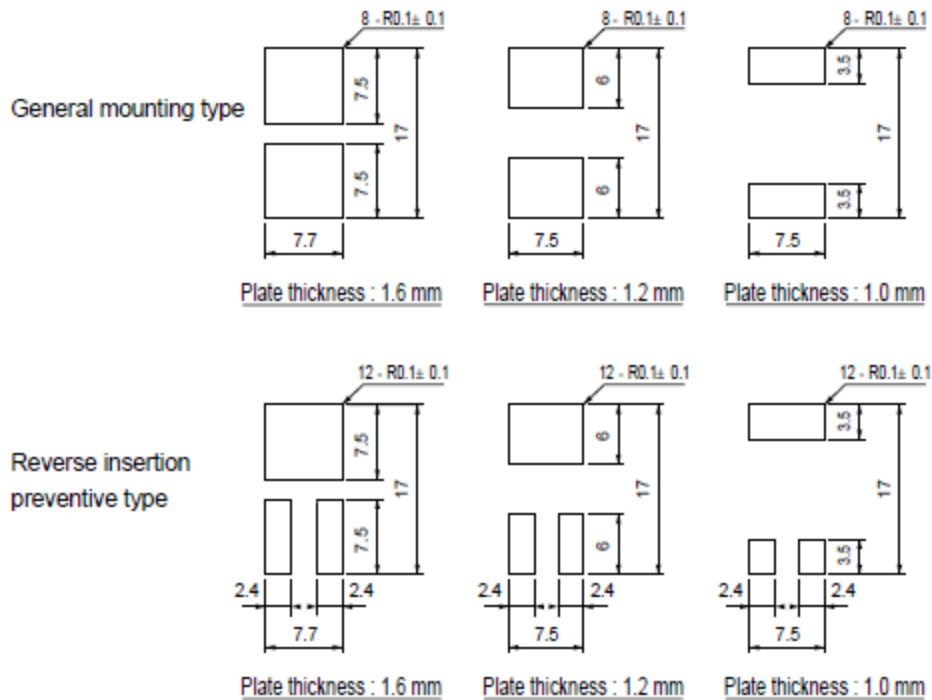


Fig. 6 Detecting Position Characteristics (2)



■ Recommended Mounting Hole Shape

(Unit : mm)



1. It is recommended to mark the shear droop surface (punch side) of the mounting plate (metal plate) with "GP1A73A" or "GP1A73A1".
2. Mounting workability, shaking after mounting and mounting strength depend on the corner radius of the mounting plate and state of punching.
Determine the mounting dimensions after check on an actual machine.
3. General dimensional tolerances shall be ± 0.1 mm.

(Precautions for Operation)

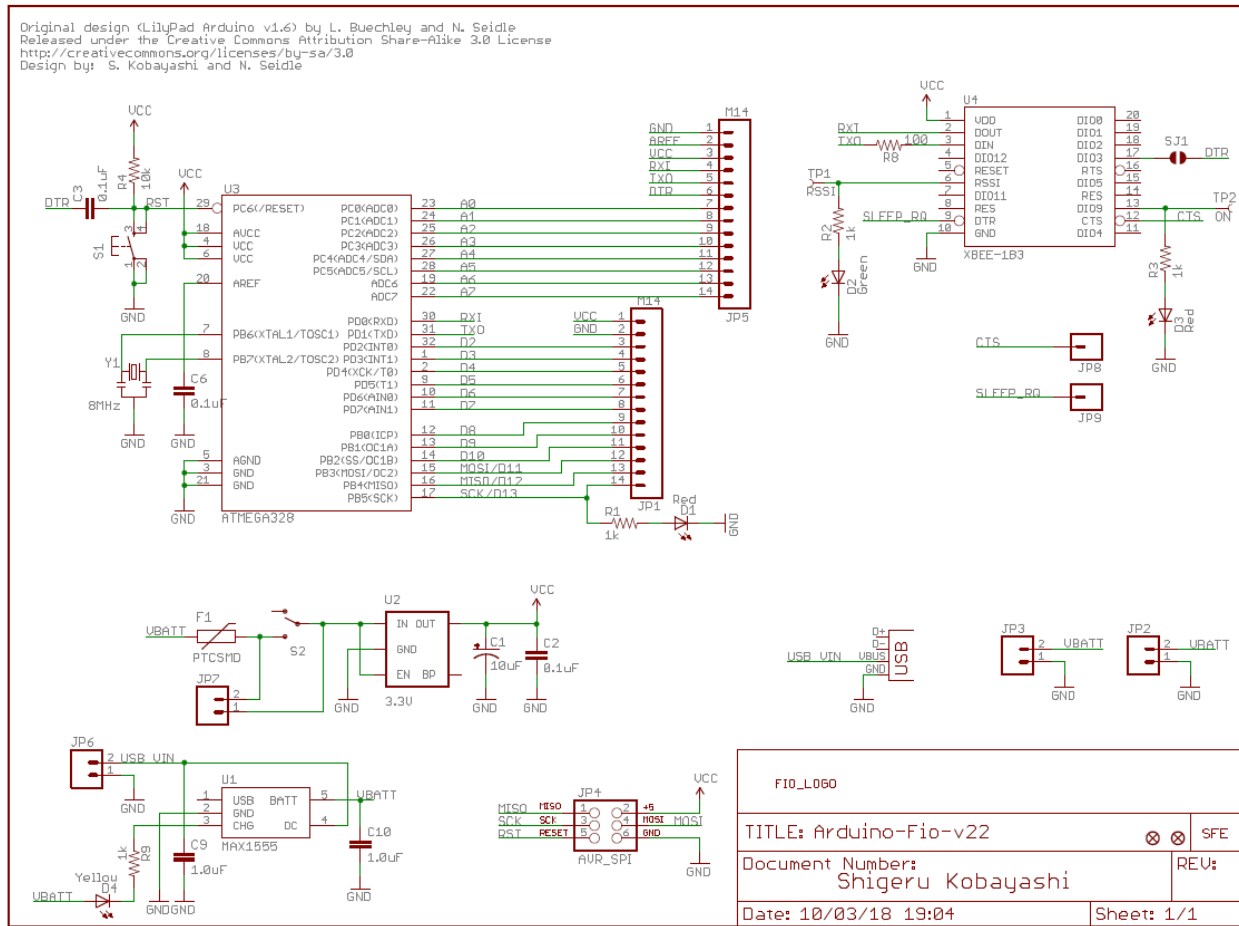
- 1) In this product, the PWB is fixed with a hook, and cleaning solvent may remain inside the case; therefore, dip cleaning or ultrasonic cleaning are prohibited.
 - 2) Remove dust or stains, using an air blower or a soft cloth moistened in cleaning solvent. However, do not perform the above cleaning using a soft cloth with solvent in the marking portion. In this case, use only the following type of cleaning solvent for wiping off;
Ethyl alcohol, Methyl alcohol, Isopropyl alcohol,
When the cleaning solvents except for specified materials are used, please contact us.
 - 3) In order to stabilize power supply line, connect a by-pass capacitor of more than $0.01\mu\text{F}$ between V_{CC} and GND near the device.
- As for other general precautions, please refer to the chapter "Precautions for Use".

NOTICE

- The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property rights. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP's devices.
- Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.
- Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:
 - (i) The devices in this publication are designed for use in general electronic equipment designs such as:
 - Personal computers
 - Office automation equipment
 - Telecommunication equipment [terminal]
 - Test and measurement equipment
 - Industrial control
 - Audio visual equipment
 - Consumer electronics
 - (ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
 - Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
 - Traffic signals
 - Gas leakage sensor breakers
 - Alarm equipment
 - Various safety devices, etc.
 - (iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
 - Space applications
 - Telecommunication equipment [trunk lines]
 - Nuclear power control equipment
 - Medical and other life support equipment (e.g., scuba).
- Contact a SHARP representative in advance when intending to use SHARP devices for any "specific" applications other than those recommended by SHARP or when it is unclear which category mentioned above controls the intended use.
- If the SHARP devices listed in this publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Control Law of Japan, it is necessary to obtain approval to export such SHARP devices.
- This publication is the proprietary product of SHARP and is copyrighted, with all rights reserved. Under the copyright laws, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of SHARP. Express written permission is also required before any use of this publication may be made by a third party.
- Contact and consult with a SHARP representative if there are any questions about the contents of this publication.

Appendix 7 – Arduino Fio Schematic

Original design (LilyPad Arduino v1.6) by L. Buechley and N. Seidle
Released under the Creative Commons Attribution Share-Alike 3.0 License
<http://creativecommons.org/licenses/by-sa/3.0>
Design by: S. Kobayashi and N. Seidle



Appendix 8 – System Block Diagram and Schematic

