# Coupler Controller Redesign

Ian Schweizer

*Department of Electronics Technology and Electrical Engineering Technology*

*University of Arkansas- Fort Smith*
*Dr. Kiyun Han*
*Ian Schweizer*

……..

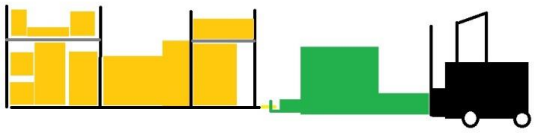
## Abstract

*The purpose of this project is to redesign the current controller system for ArcBest Technologies' Coupler controller. The Coupler is a forklift attachment that is used to move Mobile Platforms. The current controller is a physical controller that connects to a Coupler and sends commands for different functions. The current controller is typically mounted on a forklift for the operator to use. However, the controllers can be removed and are often misplaced or broken, and if multiple Couplers are present in a warehouse, it can be difficult to pair them to their corresponding controller. And, with the use of autonomous forklifts, having a physical controller to operate a Coupler is not practical.*

*The new controller system includes a new control board that actuates the hydraulics and electrical system. It is mounted on the Coupler and utilizes a mobile app that connects to the Coupler via Bluetooth to send commands to the new board. The app will be utilized in two ways; on the tablets that are currently used on the forklifts that are being used by on-site operators, and within the drive-by-wire system that the remote operators use to drive the forklifts. The new control board utilizes more efficient, reliable circuitry with a smaller footprint, helping reduce costs in both manufacturing and down-time.*

## 1. Introduction

At ArcBest, we are always researching and developing new ways to better serve our customers. One way we are doing this is by building a system that allows us to move freight within our network faster, and more efficiently. Part of this process involves a large mobile platform that can hold nearly forty thousand pounds of freight. [1.1]



[1.1]

This platform allows us to move a full trailer's worth of freight from one trailer, through an entire distribution center, to another trailer in minutes. This process normally takes hours to finish. However, forklifts cannot lift that amount of weight, so we designed and built the Coupler to mitigate this problem. When an operator needs to move a mobile platform, they first connect the forks to a Coupler [1.2], and then connect the other side of the Coupler to the mobile platform. The problems that exist within the current control system for the Coupler are Cost, ease-of-use (the remote pairing process is complicated), the ability to control more than one Coupler with a single remote, poor remote build quality, and losing remotes within large DCs. Because ArcBest has growing customer needs, more Couplers are going to be needed to ultimately move more freight. I am currently an Electrical Engineer Intern within ArcBest Technologies Research and Development and have experienced all the issues previously listed. So far, when I need to troubleshoot malfunctioning Couplers, most of the issues have to do with the remotes or control boards malfunctioning. My new control system costs five percent of what the current system does, which amounts to roughly two thousand dollars. The new design allows the app to visually replicate the layout of the current controller, reducing the time required to train an operator how to use it because it is already familiar. The physical control

board also requires less space on the Coupler [1.2], making it possible to place in more convenient locations.

[1.2]

This report will use current ABF distribution center employees as the main group example, but the usage of our Couplers is going to increase over time, so the statistics are likely to change.

For the new system to operate correctly, all you need is a charged Android device with the app installed and a Coupler. If needed, the new hardware can be mounted the same way the current board is, so there isn't a need to drill more holes. The wiring harness plug that the control board utilizes is directly compatible with the existing Coupler harness plug, allowing the new system to be truly plug-and-play. When the app is opened, a series of buttons appears on the screen that send signals to the control board when pressed. Among those buttons, there is a button that allows you to connect to a specific Coupler. Each Coupler that is discoverable will pop up in a list and is selectable by simply tapping on the Coupler name. Within the system, each Coupler can be password protected should extra security and safety precautions be necessary. Once connected, there is a specific button sequence that the operator must perform for the remote functions to activate. First, the operator must press the "ON" button, then the "SOL ON" button. After this point, all buttons are functional.

## 2. Market Analysis

The market for my design is incredibly large. Ultimately, every logistics company (or company that handles their own products) could benefit from the entirety of the system that ArcBest Technologies is developing. Because my Coupler controller is directly involved in the operation of the entire system, just the cost savings compared to the current controller would be substantial. My controller design costs 92.5% less than the system we currently use, meaning as we reach our maximum implementation capacity of approximately 900 Couplers within our own distribution and service centers, we can see cost savings easily soar above what we are currently able to achieve. That cost savings will continue to increase as we supply to more customers. To make my design an even more attractive option, the forklift operators would have an easier time using it, resulting in less time for training and down time.

As of today, there are not any other controller systems that are comparable to the one I designed other than the one we currently use, so I along with the company we use have the whole market. This is a niche market to be involved in, so there might be a few companies that contract design work for specific controls-engineering needs, however those companies wouldn't be considered as competition.

If you were to search "wireless control system," the first option that comes up are Lodar control systems. Their controllers come with a few different configurations and options, such as number of buttons on the remote, and overall capability of the system. One of the biggest features any of Lodar's control systems offer is the ability to use the controller from a remote location. Not to be confused with using the remote on-site, but by being able to send commands to the control board from anywhere in the world. There are other options that would grant the ability to control a Coupler from a remote or onsite location, such as a PLC (Programmable Logic Controller) that can connect to the same wireless network the Coupler is on. However, PLC systems are typically more expensive and when those systems need to expand or need technical support the companies that sell the PLCs make you buy other components within their product line because they are all proprietary. PLC systems are also only normally used within automation processes. My app is written entirely in Java, which is the most popular Android development language in the world. This means that it would be easy to implement in our remote operator system, and the code allows for changes to be made easily. The control board is open source and doesn't require Arduino's technical support to fix bugs or install updates. Within our distribution centers, we have more than one Coupler. Because of this, we have more than one remote to

keep track of in our inventory. Due to the way the current control board case is designed, it makes it impossible to know which Coupler can be paired with which remote without completely uninstalling the board from within the Coupler so we can see the model number on the case. Each remote has a specific model number that matches its corresponding control board. There is not a feature available that allows an operator to quickly identify which model number control board is installed on the Coupler they need to use. While there could be a spreadsheet dedicated to keeping track of which Coupler has which controller system installed, it isn't practical due to how frequently Couplers need replacement systems, and operators don't have time to sift through, potentially, pages of information to find out which Coupler they can use. My system eliminates these issues.

There are many features that my Coupler controller system shares with this competitor, but since my design focuses on both being able to be implemented within our drive-by-wire system and being much easier to use, there are also key differences that make my design a better product. Lodar is one such competitor. Most of my market research thus far has really come from all of the systems we have worked with. I have been able to directly use our current system and see the problems our operators have with it in the field. This has been the simplest way to get feedback directly from our market (wireless control systems), especially to understand what it wants to see, and to learn how we can benefit from the new controller.

## 3. Design Constraints

There were a few design constraints I considered before, and while working on my controller. Once I was given a list of requirements, I spent almost a week going over the most important factors and asking the robotics group among others within ArcBest Technologies what features they needed first and foremost. The most important design constraints/features were: Capable of implementing in remote operator systems, be more user friendly, easier to connect to Couplers, easier to know which Coupler to connect to, and be compatible with the Android devices we currently utilize.

### 3.1 End User Criteria

This was by far the most important design constraint for the controller system. I needed something that my company, ArcBest Technologies, and every other company that would want to use this, would be able to use easily. My controller is meant to be a bimodal system that can be utilized by forklift operators that are both onsite and remote. Ultimately, I knew the app would need a simple screen layout for each of the buttons that control something on our Couplers, and that the control board needed to have a smaller footprint and be easier to access. My final product is just that. Operators can easily connect to and control the Coupler they need to use. Because simplicity was one of my primary goals, not only is the software within the app easy to add to, but the buttons on the screen are easy to use with gloves.

### 3.2 Cost of Maintenance and Expansion

This was an interesting design constraint to work with, because I wasn't sure what it cost to build an app. However, the cost to produce the control board is possible to calculate because I can simply upload the Gerber files (files needed by a PCB manufacturer) and get an instant quote with approximate lead times. The cost of maintenance and expansion would be figured into the lease price.

### 3.3 Look and Feel

For any app, the look and feel are one of the most important things. When a person opens an app, they instantly judge if the layout and functions will be intuitive. For the look and feel of the app, I was not given any tips or suggestions, I simply had to design and make judgment calls myself, while referencing the design of the current handheld controller. I knew that I needed to keep the button layout as similar as possible to the current controller layout to eliminate unnecessary confusion. I also knew that I needed to make the connection process easy to perform. It is somewhat difficult to create a look and feel on a single screen while also packing in all the functions necessary to outperform and simplify the current controller. Having a solid look and feel will also help in the future when adding new features.

## 4. Design Approach

My controller system is designed with both onsite and remote forklift operators in mind. I was tasked with this project and given basic requirements. The most important focus of the app is the ease of use. There are many features within features as well, and this will be the focus of section 7, the system description.

The table below shows the design approach I have used.

| Product Design Matrix | | |
|---|---|---|
| Features | Concept A | (Current) Concept B |
| Main Page Feel | **5** | **3** |
| Ease of Use | **5** | **5** |
| Visually Appealing | **3** | **3** |
| Coupler Connection Process | **5** | **5** |
| Function Response Time | **5** | **5** |
| Power Consumption | **3** | **3** |

5 being the most desired
0 being the low priority

## 5. Relevant Professional Standards

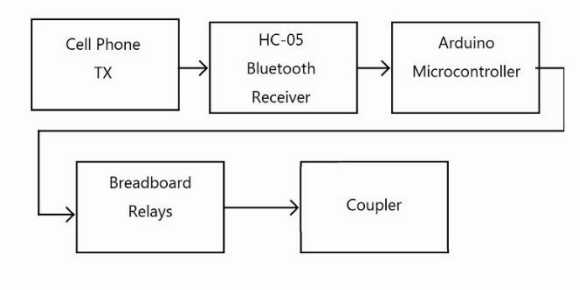
Due to the nature of the app, there will be data transmitted wirelessly. All Couplers that have the new controller board installed are password protected. When the operator selects the Coupler, they need to use in the "Bluetooth connections" dropdown menu, they are prompted for a password specific to that Coupler. The app utilizes the built-in communication protocol within the Android device which is IEEE and FCC compliant. The control board is RoHS, FCC, and CE compliant. The CE compliance is important because the control boards will likely be produced out of state, potentially in Europe.

## 6. System Description

Because my project is both an app and a physical PCB, there is focus on both the software and hardware side. The system architecture will be the layout of the app and overall system, while the hardware description will focus on the circuit design, and the software description will focus on the code. I have a simulation set up that consists of an Arduino, the bluetooth module, and two LED circuits to physically represent how the process works.

### 6.1 System Architecture

The following chart [6.21] will cover the system architecture, or rather the flow of the whole control system, and should also create a clearer understanding of what it does.

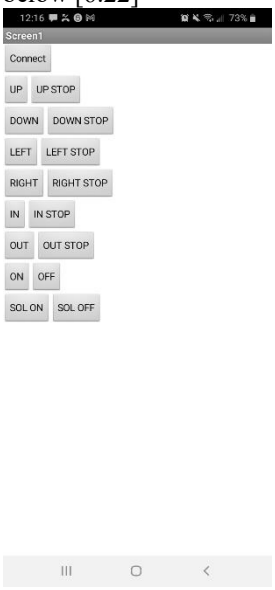



[6.21]

There are five parts to this process. The first part begins within the app on the Android device. My cell phone is the Android device used for testing; the "TX" stands for transmitter because my cell phone is transmitting the signals to the receiver. The second part is in fact the bluetooth receiver. It handles the communication between the Arduino and my Android device. The bluetooth receiver then sends the signals it receives to the third part of the process, the microcontroller. It handles all the digital logic and small signal voltage that goes to the fourth part, the relays. The relays control the high current portion of the circuit, which resides on part five, the
Coupler.

### 6.2 Hardware Description

Within the first part of the flowchart, the app exists. Technically, the app would fall within the software description, but because it is used on a physical Android device, I am including it within the hardware description as well. While there are a total of seventeen functions the app can perform via pushbuttons, they only really control eight physical functions on the Coupler. The image below [6.22]

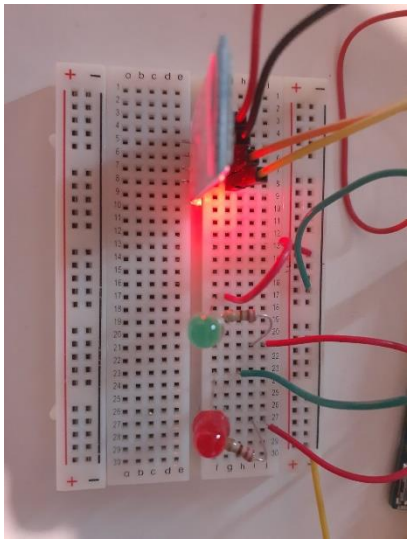



[6.22]

shows what the app looks like. The "CONNECT" button brings up a menu showing possible bluetooth device connections. [6.23]


[6.23]

As you can see, the "HC-06" bluetooth device comes up in the list. This is the bluetooth module that receives signals meant for the Coupler. It is entirely possible to rename it to fit a specific naming convention, but I left it as its default name for testing purposes. While the bluetooth module is not connected to anything, a red light will continuously flash to show that it is available to connect. Once the bluetooth module is selected in the menu, and connects, the light will stop flashing and remain on. [6.24]

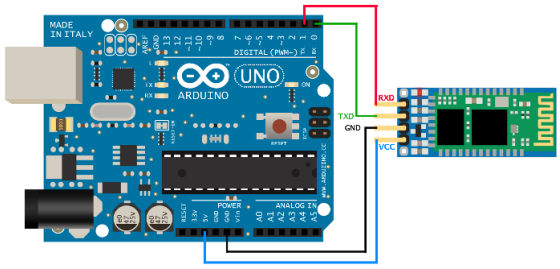


[6.24]

In picture 5, the bluetooth module is shown next to two LEDs, one being green and the other red. The built-in LED that is on the bluetooth module itself shows to be on, and not flashing. This is indicating that it is connected to the app, or the Android device, and ready to receive commands. Both the app and the bluetooth module use a serial communication protocol that essentially uses binary bits, ones and zeros, to talk to each other. In the code itself, I set each button to send the same word that it is labeled, and then convert that word to binary for the bluetooth module to receive. This way, each command is a different string of ones and zeros that can then be seen as separate commands.

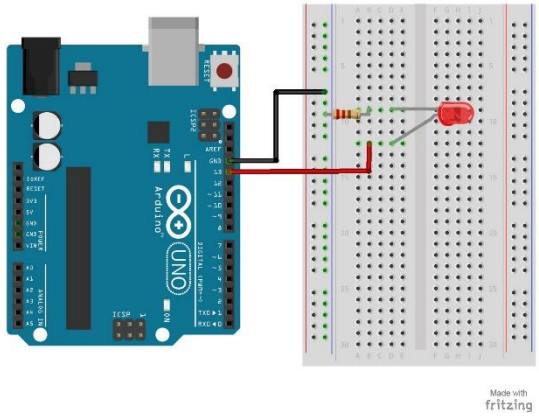
The image below [6.25], shows how the bluetooth module is wired to the Arduino board. It is simple, there are four wires. One goes to the TX (transmit) pin on the Arduino, the next goes to the RX (receive) pin, then the next two go to the five-volt power pin and GND (ground) pin. The IC

(integrated circuit) chip on the Arduino board (the long black rectangle with little silver lines on two sides) handles all the digital logic that processes the serial communication signals. Once the signals are processed (this usually takes milliseconds), the IC chip will then allow the corresponding circuit within the Arduino to become “active” or closed. Causing a specific output pin to have power.

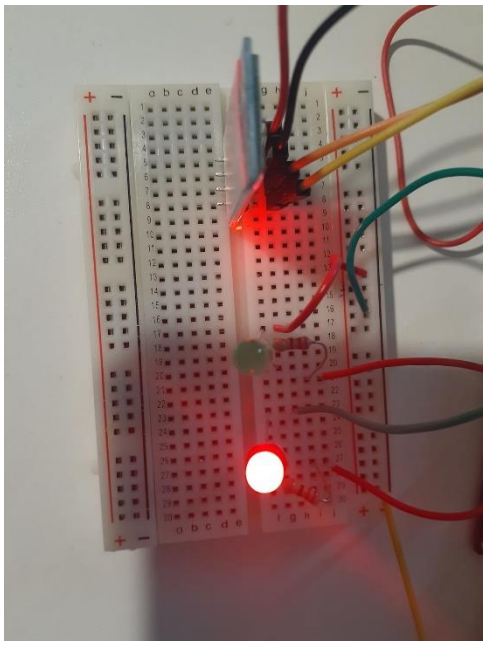



[6.25]

In this simulation, let’s say a button to turn on the red LED is pressed. The app sends binary data to the bluetooth module, the module sends it to the Arduino, then the Arduino gives power to the red LED pin. The next two pictures show what this looks like. [6.26] [6.27]

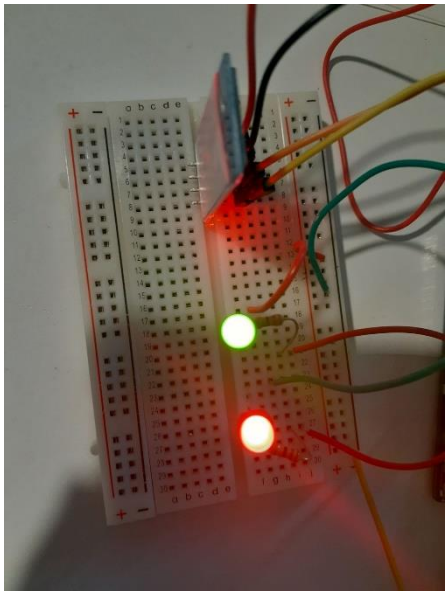



[6.26]



[6.27]

The red LED is showing what the result of this process looks like. When the operator presses a button on the app, it sends a specific command to the Coupler, and then the Coupler functions accordingly. It is important to note that there can be more than one active function at a time. The green LED shows that this is possible within my system. [6.28]

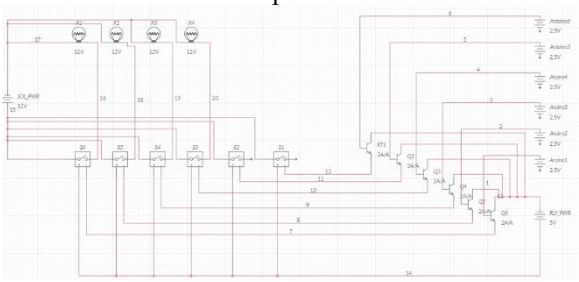


[6.28]

This is important because within the app, I included two buttons that need to be pressed every

time an operator uses a Coupler. One is to turn on the control board ("ON" button), the other is to activate a starter solenoid ("SOL ON" button). These two buttons are part of the initial button sequence I mentioned earlier that turn on portions of the Coupler that are required for operation. It is also a safety feature that requires the operator to be mindful of what buttons they are pressing.

The actual schematic for this project is more involved than the simulation version. It involves the use of multiple power supplies, BJTs (Bipolar Junction Transistor), and relays. The image below is the schematic [6.29]. I used Multisim Live to make the schematic, which UAFS has licenses to use. On the right side, the output pins are shown as individual power supplies. The transistors act as switches that are turned on by the Arduino output pins. When turned on, they complete a separate, true 5-volt circuit that powers the relays. The relays are then turned on when an internal electromagnetic coil is powered, closing the normally open side of the relay. This allows for a physical separation between the low current and high current circuits. The top left portion of the schematic shows the Coupler side.

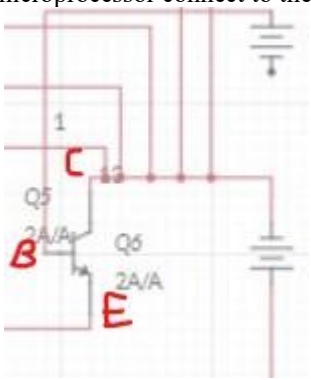


[6.29]

To choose the proper components for this circuit, each component needed to be chosen based on both mathematical computations and application principles. The foundation to the solution of every problem I faced in designing this circuit is known as "Ohm's Law." The entirety of this law can be derived from two formulas: Voltage(V) = Current(I) multiplied by resistance(R), and Power (W, P) = Current(I) multiplied by voltage(V). Once this basic principle is understood, among other more complex calculations, every variable needed to develop the above circuit can be found.

After measuring the output voltage from the pins being used on the microprocessor, I realized they were not providing a true 5 volts. Five volts is needed to turn on the relays that control the Coupler actuation. To remedy this issue, I procured an external 9-volt power source. At this point, the output pins from the microprocessor will be used to turn on the 9-volt circuits. To do this, I decided to use BJTs (bipolar junction transistors) for a couple of reasons. The first being their ability to handle the roughness that comes with an industrial environment. Second, I had more available to use compared to other types of transistors. Transistors in general have two major purposes, to act as a switch or as an amplifier. In this case, they will be used as switches.

BJTs have three pins: A Collector, Base, and Emitter. The important information regarding the BJTs within this circuit is how they are used. The 9-volt power supply's positive lead connects to all the collector pins, while the output pins of the microprocessor connect to the base pins.

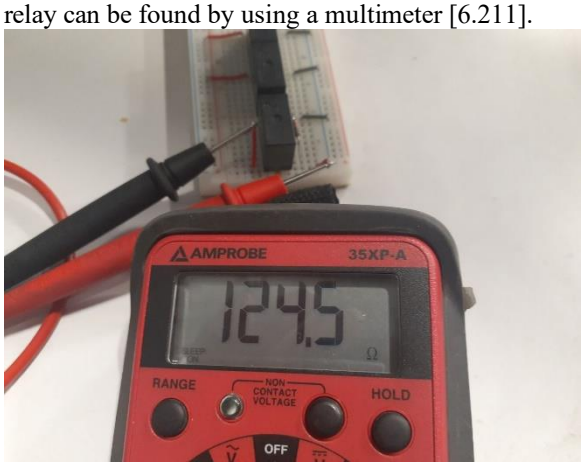



[6.210]

Think of the overall function of the BJT like a water valve. Notice the picture above [6.210]. The collector side (labeled with the red "C") is where the water comes in from the water company, the base is the handle that we rotate to control the flow of the water, and the emitter is where the water flows out of the valve. In the circuit, when the pins on the microprocessor are turned on, that voltage turns the handle, the base, to the on position. Allowing the flow of power to move from the collector to the emitter. However, there is a loss of voltage between the collector and emitter (Vce). That can be calculated by subtracting 0.7 volts from the supply voltage, 9 volts (Vc-Vbe). In total, each BJT sees approximately 8.3 volts leaving their emitters. This information is important to know because the relays only require 5 volts to be turned on. If the relays are given 8.3 volts instead of 5 volts, they could burn out sooner than anticipated. To fix this, I used the voltage divider formula that allows me to calculate voltage used by each load in a circuit.

For this to work, I need to view the relay as a resistor. The resistance of the electromagnetic coil

that is used to activate the high current side of the relay can be found by using a multimeter [6.211].




[6.211]

With this information, I calculated the size of the resistor that would consume 3.3 volts, leaving the relay with exactly 5 volts. [6.212]

$$\frac{124.5}{124.5+x}8.3 = 5$$

[6.212]

After solving for X, I get 82.17 ohms. Therefore, I needed to put an 82.17-ohm resistor in series with the emitter pin to allow 5 volts to be across the relay. Because each BJT is the same, and each relay is the same, I can put the same value resistor in series with every emitter to have a functioning circuit. Most resistors typically have a 5-10 percent allowable tolerance to compensate for actual resistance values produced during the manufacturing process, so the real PCB will need a slightly smaller value resistor to ensure the relay is getting 5 volts.

## 6.3 Software Description

The Arduino code is written within the Arduino IDE, which stands for Integrated Development Environment. It is a graphical user interface that allowed me to write the code that I used to tell the Arduino what to do for this project. The language Arduino uses is based on C/C++ but is simplified and doesn't use most of the libraries the C languages use. The following three pictures [6.31] [6.32] [6.33] help visually explain what is going on within the Arduino.

```
1  String a;
2  String b;
3  String c;
4  String d;
5  String e;
6  String f;
7  int on = 13;
8  int sol = 12;
9  int up = 8;
10 int down = 7;
11
12
13 void setup(){
14   //setup code here, to run once:
15   Serial.begin(9600);
16   pinMode(on,OUTPUT);
17   pinMode(sol,OUTPUT);
18   pinMode(up,OUTPUT);
19   pinMode(down,OUTPUT);
20 }
21
22 void loop(){
23   CoupOn();
24   CoupSol();
25   CoupUp();
26   CoupDown();
```

[6.31]

The first twenty-six lines are setting up the structure of the program. The strings at the top are telling the Arduino that it will see "messages" 'a' through 'f' come from the app. The 'int' portion on lines seven through ten indicate which pins on the Arduino are being used and assign the pins their corresponding name. For instance, pin 13 is assigned to the "ON" button. The first void module is assigning the pins as output pins, while the second void module contains the following modules that utilize the command signals from the app. It allows the Arduino to call the individual module needed faster.

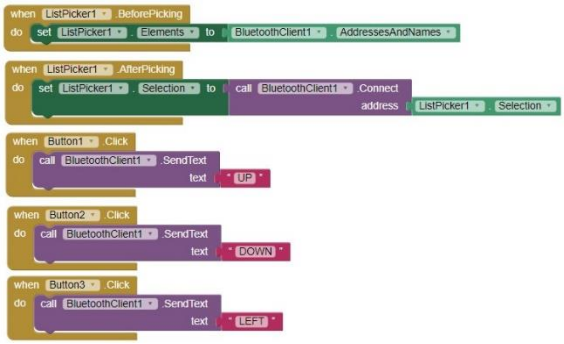
```
25   CoupUp();
26   CoupDown();
27 }
28   //put main code here, to run repeatedly:
29  void CoupOn(){
30   while(Serial.available()){
31     a=Serial.readString();
32     Serial.println(a);
33     if(a=="ON"){
34       digitalWrite(on, HIGH);
35     }
36     if(a=="OFF")
37     {
38       digitalWrite(on,LOW);
39     }
40   }
41 }
```

[6.32]

The module shown above shows how the message ‘a’ is read. Within ‘a’, there can be serial data that says “ON” or “OFF”. If the data reads “ON”, it turns the “on” pin, pin 13, to the on state. When it is in the “HIGH” state, it means it has power. There are four voids, or modules, within the Arduino program that operate exactly as seen in picture [10]. Instead of saying “CoupOn”, they start with “CoupSol”, “CoupUp”, and “CoupDown”.

The Android app is written in Java, primarily in Visual Studio. It is much more involved than the microprocessor program. In total, the microprocessor program consists of around 80 lines, while the app consists of nearly 850 lines. The primary reason for this is layout of the GUI (graphical user interface) requires a lot of code. The code that handles the bluetooth communication is also more involved.

To start, I used a tool called MIT App Inventor, which utilizes block coding to create both Android and iPhone apps. This allowed me to visually see how I needed to structure my program. [6.33]



[6.33]

The blocks (yellow) help organize the code in a way that makes it easier to understand. The first two handle the bluetooth bootstrapping, which means the app is going to use the existing bluetooth capabilities already built into the Android device to send serial data to the control board. The following yellow blocks show how each button works. When pressed, they send text that is specified in the red box. Each text is different, allowing each button to send their own command.

This tool was especially helpful because I had never made an app until this project. Because I started with zero experience, I needed extra help in the beginning stages.
Once I finished the structure of the app, I was able to start creating the Java source code. The following pictures show portions of the code that have to do with a button [6.34], and the bluetooth portion [6.35] [6.36].

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_marginTop="10dp"
    android:layout_height="wrap_content">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.1"
            android:text="RX:"
            android:ellipsize="end"
            android:maxLines="1"
            android:textStyle="bold" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.9"
            android:ellipsize="end"
            android:maxLines="1"
            android:text="<Read Buffer>"
            android:id="@+id/readBuffer"
            android:layout_centerHorizontal="true" />
    </LinearLayout>
```
[6.34]

In picture [6.34], a portion of the structure of one of the buttons is shown. This shows the amount of code needed to simply determine the size and place of a button. In picture [6.35], the app is required to receive permissions from the bluetooth function already in the Android device. Once it receives those permissions, it is free to use the bluetooth capabilities, also known as being able to bootstrap.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<permission android:name="android.permission.BLUETOOTH" android:label="BLUETOOTH" />
<permission android:name="android.permission.BLUETOOTH_ADMIN" />
<permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```
[6.35]

The following code shows a small portion of the bluetooth handler. This is the portion of the app code that simply tells the user if they successfully connected to the bluetooth module on the control board.

```
mHandler = new Handler(){
    public void handleMessage(android.os.Message msg){
        if(msg.what == MESSAGE_READ){
            String readMessage = null;
            try {
                readMessage = new String((byte[]) msg.obj, "UTF-8");
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
            mReadBuffer.setText(readMessage);
        }

        if(msg.what == CONNECTING_STATUS){
            if(msg.arg1 == 1)
                mBluetoothStatus.setText("Connected to Device: " + (String)(msg.obj));
            else
                mBluetoothStatus.setText("Connection Failed");
        }
    }
};
```

[6.36]

If the connection is successful, the app simply drops the available device menu. If it was not successful, it will tell the operator that there was an error to connect [6.37].


[6.37]

## 7. Theoretical Analysis

Because of my controller design, we can go from a controller that can only be used onsite at a distribution or service center, to a system we can use both onsite and from remote locations. This new feature amplifies our capability of being more cost effective and gives forklift operators the choice to work in a comfortable, climate-controlled environment. The ease of use and architecture of the design gives the operator the ability to be more efficient while working and allows for implementation within our drive-by-wire system. The improvements my design brings to the use of our Couplers allows our customers to not only save money both short and long term, but it also creates a safer environment onsite.

1. Ease of use
    a. App resides on Android device, no need to keep up with multiple remotes
    b. Device pairing process is much simpler and more intuitive, taking less time to connect and allows for faster movement of freight.
2. Drive-By-Wire Utilization
    a. App software can be implemented within existing remote forklift operator systems.
    b. I control the source code, allowing for on demand updates and do not have to rely on third party support.
    c. Design Principles
    d. Simple, familiar, easy to use. Similar look to the current remote, which cuts down on training time.
    e. LTC-Legible Text Content. Buttons are easy to read and press.

## 8. Validation and Testing

My validation and testing have been done by myself. I have been working with Couplers that I have access to within our R&D facility. I have taken into consideration the complaints I have heard from coworkers regarding the current Coupler control system, along with use case issues from the field. I have installed the app on my personal cell phone and have tested each function the Couplers have. I designated a test Coupler for my control system that makes sure all the functions operate correctly. All the functionality is, of course, functioning. I will also continue testing my design on other Couplers utilizing the quick connect plug to ensure there aren't any faults or edge case issues—which happens to be the best form of testing for my project.

## 9. Results and Conclusion

I started my project in January of 2022 and have come a long way since then. I started with an idea and created a control system that will be a tool that my company can use for a long time. As of now, the app may not be as pretty as I would have liked, but all the functionality required is there. I will continue to work with the people involved to add features that my bosses and other operators want, and I will continue to work with others and get new ideas. My goal for the future is to design a PCB (printed circuit board) from the ground up that is specifically designed for this system. That way it will be more efficient, have a smaller footprint, and hopefully cost less to produce. I would also like to add Wi-Fi capability to the control board, so the remote operation of the Coupler can be feasible. I would also like to add an error code feedback system that automatically sends error messages to both the operator and supervisor with specific information regarding any problems that occur. Within that system, I would like to add a "quick fix" troubleshooting guide to allow technicians to solve any issues faster without it being necessary to send the Coupler or control system back for in depth troubleshooting.

## 10. References


[1] "Lodar Industrial Wireless Control System," *Pierce Arrow Inc.*, 21-Apr-2022. [Online]. Available: https://www.piercearrowinc.com/product/industrial-wireless-control-system/. [Accessed: 24-Apr-2022].

[2] "United States Coverage Area," *ArcBest*. [Online]. Available: https://arcb.com/coverage-area/us-shipping. [Accessed: 24-Apr-2022].

[3] "MIT App Inventor | Explore MIT App Inventor," *Mit.edu*, 2019. https://appinventor.mit.edu/

[4] Multisim Live Online Circuit Simulator, "Multisim Live Online Circuit Simulator," *NI Multisim Live*, 2019. https://www.multisim.com/

[5] Microsoft, "Visual Studio," *Visual Studio*, 2019. https://visualstudio.microsoft.com/

[6] "Software," *www.arduino.cc*. https://www.arduino.cc/en/software


## 11. Appendix

### 11.1. Project Contract (Statement of Work) Section I: Statement of Work

Date: 01/27/2022
Client: ArcBest Technologies
Name: Coupler Control System Redesign
Requested By: ArcBest Technologies
From: Ian Schweizer

**Section II: Summary/Purpose**

To provide ArcBest Technologies with a way to implement their pre-existing Coupler control system into their remote forklift operator framework. I will build an app and redesign the control board. This design will fulfill the needs of ArcBest Technologies, along with both onsite and remote forklift operators.

**Section III: Project Scope**

The scope of this project is to utilize knowledge gained in the Electrical Engineering Technology program at UAFS. In this project, I will program an Android app to connect and send commands to a PCB that I will design and have manufactured. The general functions of the PCB will be to communicate via bluetooth with an Android device using the app, then the PCB will have hard-wired connections to the actuation of the Coupler. The app will be developed using both MIT App Inventor and Visual Studio Community 2021. The PCB will be designed using Multisim Live, EasyEDA, and Altium Circuit Maker.

**Section IV: Location/Schedule**

I have been allotted approximately twenty percent of my time at work to spend on this project. All other time spent must be on my own time.

### 11.2 Code of Conduct

I am the sole member pertaining to this project and will be doing all the coding and electronics/systems engineering design work.

Tentative schedule:

1. January end

a. Have code written for Arduino prototype

2. February end
   a. Schematic done, begin layout of app.
3. March end
   a. App done, begin testing communications.
4. April end
   a. Done with testing and Change Requests

**Section V: Applicable Standards**

Standard password protection will be used to ensure only authorized users make connections with Couplers. Data encryption will be used to protect any serial data transmissions from being stolen or misused.

**Section VII: Acceptance Criteria**

To ensure the application is acceptable during the project I will continually receive, and ask for, feedback from ArcBest Technologies while testing the control system. I will align the features and changes to their needs.

Under these conditions I will continue to improve and change the control system. The needs of the application, however, will continually evolve. When changes to the existing design have been approved and committed by both the client and I, I will add it to the items required for acceptance. Once the control system functions to specification according to the iterations agreed upon by myself and the client then I will continue to work on new iterations.

**Section VIII: Acceptance**

Organization: ArcBest Technologies
Full Name: ArcBest Technologies
Title: Company

Organization: Ian Schweizer
Full Name: Ian Schweizer
Title: Sole Team Member

I plan to work diligently and make sure that I finish with the best project and product possible. I agree to:

· Meet deadlines
· Work as hard as possible and not cut corners
· Make sure I set aside time for the project even with my busy schedule
· Receive constructive criticism
· Share what I am working on with ArcBest Technologies to receive feedback
· Track everything I do
· Use Visual Studio 2021 and all its capabilities to the fullest
· Focus on the project when necessary and not become distracted
· Work ahead, such as on future assignments and presentations that I know are coming up
· Have fun!

I also agree to ask for any help that I need within ArcBest Technologies' robotics team. Any new issue that arises I will handle or ask for help and decide the best course of action as I want this project to impress and be the best that it could be.

1/27/2022
Ian Schweizer

"I have read the entire report and it meets my personal quality standards."
Ian Schweizer