TEAM THUMBSTART

A keyless ignition system using a fingerprint scanner for motorcycles

Brandon Aldecoa
Robert Rebich
Anas Husain
Table of Contents

1. Background ......................................................................................................................1

2. System Architecture ......................................................................................................2

3. Concepts Considered ......................................................................................................4
   3.1 Fingerprint Scanner ...................................................................................................4
   3.2 Radio Frequency Link ...............................................................................................4

4. Concepts Chosen ............................................................................................................5
   4.1 Fingerprint Module .................................................................................................5
   4.2 Microcontroller .......................................................................................................6
   4.3 Transmitter ..............................................................................................................7
   4.4 Receiver ..................................................................................................................8
   4.5 Encoder/Decoder .....................................................................................................9
   4.6 ATTINY45/Relay ....................................................................................................10

4. Future Work ..................................................................................................................11

Appendix A .......................................................................................................................13
   Table 1. Backup system decision matrix .......................................................................13
   Table 2. Budget breakdown ............................................................................................13
   Table 3. List of part for transmitter circuitry .................................................................14
   Table 4. List of part for receiver circuitry .....................................................................15

Appendix B .......................................................................................................................16
   Figure 8. Transmitter schematic ...................................................................................16
   Figure 9. Receiver schematic .......................................................................................17

Appendix C .......................................................................................................................18
   Final board layout

Appendix D .......................................................................................................................19
   Code for ATMEGA48 in transmitter circuit

Appendix E .......................................................................................................................29
   Code for ATTINY45 in receiver circuit

Important Data Sheets ....................................................................................................30
1. BACKGROUND

Biketronics Inc, a local company in Moscow Specializing in motorcycle electronics requested the help of a senior design team from the University of Idaho. Team ThumbStart has been chosen to research and design a fingerprint scanner system that can replace the traditional key ignition system that is commonly used on motorcycles. The integration of a keyless ignition system using a fingerprint scanner into existing motorcycle technologies is possible. There are many different design considerations that have been taken into account. We have proposed a design and have a working final prototype that can successfully implement the fingerprint system into a wide variety of motorcycles.

This system uses fingerprint data and allows the vehicle owner to operate the motorcycle without the need of keys. The system requires no hardware to be carried, but must maintain the same or higher level of security than the key ignition. The design only allows authenticated users to enroll a new fingerprint as well as start and operate the bike. The system will not activate if an authorized user is not authenticated. A method to train at least two sets of unique fingerprint has been implemented. Bikes have very little open space to store things. The entire package is small enough to fit on a wide variety of motorcycle designs. The final product must be reliable enough to withstand the harsh conditions of the road. The final product will be backed by a lifetime company warranty, which means the final design must be reliable and robust.

This project consists of combining of the shelf technologies together in a new way that can be beneficial to motorcyclists and possibly others. Currently, all the main hardware components of our project have been developed by companies and are readily available for purchase. The circuitry and algorithm for the Suprema fingerprint scanner has been purchased in a stand-alone module. Other major components such as the ATMEL
microprocessor and the keypad are also available and have been purchased. A large portion of the overall design is coding an algorithm to implement the desired functions between each hardware block. Other major design criteria are the environmental conditions that the system will be exposed to. This means that the device will be exposed to rain, heat, cold, grease, dirt and many other harmful and possibly damaging factors. The device is designed so it can be mounted almost anywhere on the motorcycle. It is also small enough so it can be hidden under other parts of the motorcycle. The integration of this design required knowledge of laboratory equipment, C programming, circuit level board design and chip level communication.

2. SYSTEM ARCHITECTURE

![System Architecture Diagram]

Figure 1.
System Architecture

Above is a top-level block diagram of the system architecture. The design is relatively simple composing of five main components. The main components consist of a microcontroller, a fingerprint scanner module, a keypad, a radio frequency (RF) linkage, and
a relay that is controlled through another microprocessor. Some of the main components require external lower-level circuitry for practical operation, which is not depicted in the top-level diagram.

The heart of the system is the microcontroller that is capable of handling communications between the fingerprint scanner module, keypad and transmitter. All operations are controlled through the microcontroller. The microcontroller stores the main C program onboard, which is responsible for the entire system operation. Nothing will function independently of the microcontroller.

The fingerprint scanner module is responsible for detecting the thermal signature of the fingerprint into a usable template image. The onboard computer determines if the scan was valid or not and relays that information to the microcontroller. There is two-way communication between the microcontroller and the fingerprint scanner module.

The keypad is the only input to the system that the user will have. The keypad is connected directly into the microcontroller, which processes all button inputs. Every system will come with a default code. One first use the user must reset the code to ensure the code is unique to their system. This user code unlocks the system so that new fingerprint can be added to the system.

The RF linkage takes place between the microcontroller and the motorcycle’s starter relay. The RF block consists of an encoder and transmitter pair and a receiver and decoder pair. The microcontroller communicates through the transmitter, letting the receiver know that a valid scan occurred. The encoder and decoder are used to convert a digital signal into a RF signal and also to encrypt the RF transmission using KEELOQ. When a valid scan occurs the microprocessor on the receiver will engage the relay.
3. CONCEPTS CONSIDERED

Among many different ideas that were considered over the scope of the project the following are the major considerations that we took into account when selecting the major components of the project.

3.1 Fingerprint Scanner

There are many different options for fingerprint scanners that could have met the needs of this project. We researched various types of scanners, their reliability, and temperature range. Suprema Inc. offers many different solutions with products that offer thermal, optical, capacitive or electromagnetic scanners. Each scanner had its benefits and downsides whether it was fingerprint recognition speed, temperature range, or scanner footprint. All of these factors were taken into consideration when selecting a scanner.

Some of them came with DSP that had the fingerprint algorithms already programmed onto them while others just offered the scanner as a stand-alone module. We looked at some of the common algorithm methods used for pattern recognition. Though some seemed manageable to possibly implement this seemed unnecessary when there were products that already had the algorithm implemented with the scanner.

3.2 Radio Frequency Link

There are many different companies that produce small RF receivers and transmitters. We were looking for a simple, reliable, easy to implement transmitter and receiver that would meet our needs. Micrel Inc. produced a wide variety of transmitters and receivers. They have simple transmitters that are similar to garage door openers to transmitters that can transmit serial data and all the matching receiver packages. A low cost
and low power consumption unit that can transmit an on off signal are going to be the main factors for our selection.

Another option is also available. A company produces wireless relays that are turned on using a key fob. We could use this to be our RF link to a relay. The draw back to this is that it is much more expensive. A circuit designed and built by us will be about a quarter of the cost of the wireless relay produced by this company.

4. CONCEPTS CHOSEN

4.1 Fingerprint Module

A company called Suprema Inc offers a stand-alone embedded fingerprint module. The module is comprised of scanner that is controlled directly through a digital signal processor (DSP).

Suprema offers four different types of scanners that can be attached to the DSP. The thermal scanner has many advantages over the other scanner technologies. The thermal scanner is very simple with a robust design that can withstand the road conditions. The thermal scanner naturally cleans itself after each swipe, making it ideal for the situation.

The DSP contains a built in fingerprint recognition algorithm. The algorithm can process a single fingerprint in less than one second and has the ability to read finger swipes from up to a 45-degree finger rotation. The on-board memory can store almost two
thousand different fingerprint templates, which allows more than enough users to be enrolled into the system.

The orange ribbon cable that connects the DSP with the scanner is only a few inches in length. The cable cannot be lengthened because of the serial communication that happens on the traces. If the cable is longer, the serial communication may become unrecognizable from cross talk and other forms of distortion. The two components are packaged together. The thermal scanner is folded over the bottom of the DSP and mechanically mounted together.

4.2 Microcontroller

There are many different types of microcontroller available for purchase. Biketronics has been using a microcontroller from Atmel in many of their other products. The decision to use an Atmel microcontroller was strongly suggested because of their wide range of chips that vary in memory size, capability and speed.

![ATMEGA48 Layout](image)
The ATMEGA48 microprocessor from the Atmel AVR family is chosen because it satisfied every need of the design. The 28-pin package is small in size, has industrial temperature ratings and can handle up to 23 input-output pins (I/O’s). Although not all the I/O’s are used, further expansion if the design can be easily implemented. The chip also has two forms of onboard memory: 4Kb Flash and 256b EEPROM. The C code is stored into Flash memory and the unique user code is stored into the EEPROM. The chip also supports serial communication. The universal synchronous asynchronous receiver/transmitter (USART) serial protocol is used to communicate between the fingerprint scanner DSP and the microcontroller. The onboard RC oscillator on the microcontroller cannot be calibrated well enough to maintain precise USART communication; an external crystal oscillator is used for the precise clocking. The operating voltage of this chip is between 3 and 5 volts. The fingerprint scanner module operates at 3.3 volts; the Atmel chip is also being operated at the same potential. The pins on the microcontroller supply up to 40mA of output current that are used to supply onsite power to other board components.

4.3 Transmitter

A transmitter is needed to send the radio frequency transmission to the starter relay device to engage it. Micrel Inc provides a wide variety of transmitters for easy implementation. The MICRF113 is the ideal transmitter for our needs and is the chosen model. This transmitter has an operating voltage range from 1.8 to 3.3 volts and very low current consumption, typically 12.3mA. The device powers from an IO pin on the microcontroller so that the device is only consuming power when there is a transmission. This keeps the power consumption of the entire system to a minimum. External resistors, capacitors and inductor are used to build a matching network for the antenna.
The MICRF113 transmits at a range of frequencies depending on the value of the crystal oscillator as well as capacitors connected to pins 1 and 4. We set our transmitter to transmit at 315MHz using a 9.84375MHz crystal oscillator. It operates with amplitude shift keying/on-off keyed protocol (ASK/OOK). This transmitter transmits up to 200m if properly matched to the antenna. For this application, transmission is a maximum of 5m so antenna matching is not important and a simple wire antenna is used.

The small 6-pin SIOC package is the simplest transmitter produced by Micrel. In the pin layout the ASK pin is the input data pin. The oscillator and capacitors are connected across pins XTLIN and XTIOUT. The PAOUT pin is connected to the antenna.

4.4 Receiver

Micrel also produces a wide variety of receive modules that are compatible with their transmitter. The MICRF010 we selected on recommendation from a technical representative from Micrel. This is also a small 8-pin SIOC package that operates at 5V. This part also runs on a very low current, 2.9mA typically. The receiver is highly sensitive (~104dBm) and has automatic tuning so there is no need to manually do this. An oscillator, capacitors and inductors are used to set the center frequency at 315 MHz to match with the transmitter.

The oscillator is connected to the REFOSC pin. The frequency is set by the values of inductors and capacitors connected to CTH, SHUT, and CAGC. The antenna on the receiver is connected to the ANT pin and again no matching between the antenna and chip.
because the short distance these devices are transmitting over. The DO pin is the data out pin. The data put into the ASK pin on the transmitter is the output on the DO pin on the receiver package.

### 4.5 Encoder/Decoder

Biketronics will be producing many of these fingerprint scanner systems. There needs to be a way to pair one transmitter to one receiver for security reasons so that a receiver will engage the relay only when it receives a transmission from its paired transmitter. An encoder and decoder are used to ensure pairing between only the two devices. Microchip sells a variety of encoder and decoders that are very inexpensive; each unit is less than $2.00.

This encoders and decoders use a protocol called KEELOQ to encrypt and decrypt data. The encoder we are using is the HCS300. It has an input voltage of 3.3V and also draws very low current when it is idle. This encoder has four input pins that are pulled high or low for input states. These pins are connected to the ATMEGA48 microprocessor. The KEELOQ protocol encrypts these into a different unique 64-bit packet that it sends out on the PWM line. Even for the same input the 64-bit packet is unique for every transmission. The PWM pin is connected to the ASK input pin on the transmitter.
The decoder, HCS515, is connected to the MICRF010 receiver. This also operates at 5V just like all the circuitry in the receiver module. The DO pin on the receiver is connected to the RF_IN pin on the decoder. This data is still encrypted and the KEEOQ protocol decodes the data stream and then outputs it on pins S0 and S1.

The data has now traveled from the microprocessor, been encrypted, transmitted, received and then decrypted again. The HCS515 is not the smallest decoder Microchip manufactures but it is the smallest chip that has the data out pins S0 and S1. The smaller modules only output serial data, which is more complex and unnecessary for our application.

An encoder and decoder are paired together. There are several different pairing techniques and we are using the most basic pair method. To do this we have purchased a KEEOQ Evaluation Kit. The decoder is put into a learn mode and then the encoder sends it a data stream. The decoder then knows the unique seed for the encoder and the two devices are paired and only work with each other.

4.6 ATTINY45/Relay

Each motorcycle has several relays that control different electronics, lights or ignition of the bike. We modified the relay that controls the ignition so that a microprocessor controls when the relay is engaged. The microprocessor selected was the Atmel ATTINY45 due to the small package. There are 8 pins on the package and only 6 I/O’s. This is the smallest package Atmel produces.
When the receiver receives a good scan signal, an input pin is pulled high on the ATTINY45 for 50ms. The ATTINY45 then pulls an output pin high turning on a N-channel enhancement mode MOSFET. The code on the microprocessor is very basic. It loops and waits for an input on one pin and when it gets an input on that pin it pulls a different pin high to turn on the MOSFET.

The MOSFET is connected and used as the switch in Figure 7 to turn on the relay. The gate of the MOSFET is connected to the output from the ATTINY45, the drain is connected to relay and the source is connected to ground. The relay will not be able to be turned on until the MOSFET is turned on.

5. FUTURE WORK

A working prototype of this product was produced and confirmed by starting a motorcycle. The user was able to set a new unique user code, enroll new fingerprints, delete all the fingerprints currently on the system and most important start the motorcycle. Though the prototype did function areas of the design still need to be improved before it can be sold as a reliable final product.

The microcontrollers on both the receiver and transmitters have working code. The code for ATMEGA48 microcontroller on the transmitter needs to be expanded. The user needs to have a user code they program into the system when they first receive it. This user code needs to be stored to EEPROM so that it will not be lost when power to the system is
lost. The menu for a user to input a unique code is in place it only needs to be modified to store the value into the non-volatile memory.

The RF link is functional and has shown to have a much greater range than is necessary for this application. When testing the RF we found that it is much more sensitive than we expected. The parts had to be handled with care not to damage any of them. For this reason a package that is very protective for the circuit is extremely necessary to minimize the amount of vibration, temperature swings, and overvoltage to the part. If these items are taken into consideration for the packaging the device should operate properly over the short distance range.

The packaging for the entire transmitter module must also be considered. Though this module has shown to be more rugged than the receiver module this is still subject to temperature and other weather conditions. The enclosure holding the scanner module as well as the other transmitter circuitry also needs to be aesthetically pleasing on a motorcycle.
Appendix A

<table>
<thead>
<tr>
<th>Part</th>
<th>Cost</th>
<th>Order size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel AVR PIC</td>
<td>$2.50</td>
<td>1</td>
</tr>
<tr>
<td>Suprema SFM3010-FC</td>
<td>$125.00</td>
<td>1</td>
</tr>
<tr>
<td>Keypad</td>
<td>$10</td>
<td>1</td>
</tr>
<tr>
<td>Relay</td>
<td>$15</td>
<td>1</td>
</tr>
<tr>
<td>Oscillators</td>
<td>$1.50</td>
<td>1</td>
</tr>
<tr>
<td>All LED's, Capacitors, and Inductors</td>
<td>$5.00</td>
<td>20 of each</td>
</tr>
<tr>
<td>Voltage regulators</td>
<td>$0.78</td>
<td>20</td>
</tr>
<tr>
<td>MICRF010 Receiver</td>
<td>$2.21</td>
<td>10</td>
</tr>
<tr>
<td>MICRF113 Transmitter</td>
<td>$0.88</td>
<td>10</td>
</tr>
<tr>
<td>HCS300 Encoder</td>
<td>$1.63</td>
<td>10</td>
</tr>
<tr>
<td>HCS515 Decoder</td>
<td>$3.35</td>
<td>10</td>
</tr>
<tr>
<td>Total Cost</td>
<td>$152.85</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1
Budget

<table>
<thead>
<tr>
<th>Key fob (RF)</th>
<th>Keypad</th>
<th>Handle bar combo</th>
<th>Smart card</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pros</td>
<td>Wireless</td>
<td>No hardware to carry</td>
<td>No external device needed</td>
<td>Cheap, Small, light</td>
</tr>
<tr>
<td></td>
<td>· Already using RF</td>
<td>· Cheap, simple</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cons</td>
<td>· Need to have device with you</td>
<td>· Extra piece of hardware to implement</td>
<td>· Complicated to implement</td>
<td>Smart card required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Near impossible to implement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2
Cost breakdown of various components
<table>
<thead>
<tr>
<th>Part name</th>
<th>Value</th>
<th>Unit</th>
<th>Package/Footprint</th>
<th>Manufacturer</th>
<th>Digi-key Part number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suprema</td>
<td>N/A</td>
<td>N/A</td>
<td>SFM 3010-FC</td>
<td>Suprema INC</td>
<td>N/A</td>
</tr>
<tr>
<td>Abracron</td>
<td>1.8432</td>
<td>MHz</td>
<td>HC49/U</td>
<td>Abracon Corporation</td>
<td>535-9009-ND</td>
</tr>
<tr>
<td>C1</td>
<td>22</td>
<td>uF</td>
<td>Threw hole 1.00mm</td>
<td>Panasonic - ECG</td>
<td>P950-ND</td>
</tr>
<tr>
<td>C1_T</td>
<td>10</td>
<td>uF</td>
<td>_0805</td>
<td>muRata</td>
<td>490-1717-1-ND</td>
</tr>
<tr>
<td>C10_T</td>
<td>0.1</td>
<td>uf</td>
<td>_0603</td>
<td>muRata</td>
<td>490-1519-1-ND</td>
</tr>
<tr>
<td>C13_T ,C14_T</td>
<td>18</td>
<td>pF</td>
<td>_0603</td>
<td>muRata</td>
<td>490-3573-1-ND</td>
</tr>
<tr>
<td>C2</td>
<td>10</td>
<td>uF</td>
<td>_0805</td>
<td>muRata</td>
<td>490-1717-1-ND</td>
</tr>
<tr>
<td>C2_T</td>
<td>100</td>
<td>pF</td>
<td>_0603</td>
<td>muRata</td>
<td>490-1427-1-ND</td>
</tr>
<tr>
<td>C3 , C4</td>
<td>18</td>
<td>pF</td>
<td>_0603</td>
<td>muRata</td>
<td>490-3573-1-ND</td>
</tr>
<tr>
<td>C5_T</td>
<td>10</td>
<td>pF</td>
<td>_0603</td>
<td>muRata</td>
<td>490-3570-1-ND</td>
</tr>
<tr>
<td>C7_T</td>
<td>6.8</td>
<td>pF</td>
<td>_0603</td>
<td>muRata</td>
<td>490-3563-1-ND</td>
</tr>
<tr>
<td>Encoder</td>
<td>HCS300</td>
<td>Encoder</td>
<td>8-SOIC</td>
<td>Microchip Technology</td>
<td>HCS300/SN-ND</td>
</tr>
<tr>
<td>I_lim1,2,3</td>
<td>147</td>
<td>Ohm</td>
<td>_0603</td>
<td>Panasonic - ECG</td>
<td>P147HCT-ND</td>
</tr>
<tr>
<td>IC1</td>
<td>MIRCF113</td>
<td>Transmitter</td>
<td>SOT23-6</td>
<td>Micrel Inc</td>
<td>576-3229-1-ND</td>
</tr>
<tr>
<td>L1_T</td>
<td>470</td>
<td>nH</td>
<td>_0805</td>
<td>Toko America Inc</td>
<td>LLQ2012-ER47J-ND</td>
</tr>
<tr>
<td>L4_T</td>
<td>150</td>
<td>nH</td>
<td>_0603</td>
<td>JW Miller A Bourns Company</td>
<td>M1257CT-ND</td>
</tr>
<tr>
<td>LM340MP-05</td>
<td>5</td>
<td>V</td>
<td>SOT223</td>
<td>Micrel Inc</td>
<td>MIC2920A-5.0BS-ND</td>
</tr>
<tr>
<td>MEGA-48</td>
<td>ATMEGA48</td>
<td>N/A</td>
<td>28-DIP</td>
<td>Atmel</td>
<td>ATMEGA48-20PI-ND</td>
</tr>
<tr>
<td>Q4 - MOSFET</td>
<td>N-Channel</td>
<td>N/A</td>
<td>SOT 23-3</td>
<td>STMicroelectronics</td>
<td>497-3111-1-ND</td>
</tr>
<tr>
<td>R2_T</td>
<td>100</td>
<td>kOhm</td>
<td>_0603</td>
<td>Vishay/Dale</td>
<td>541-100KHCT-ND</td>
</tr>
<tr>
<td>Y1</td>
<td>9.84375</td>
<td>MHz</td>
<td>HC49/US</td>
<td>Abracon Corporation</td>
<td>535-9689-1-ND</td>
</tr>
</tbody>
</table>

Table 3

Parts List for the Transmitter side
<table>
<thead>
<tr>
<th>Part name</th>
<th>Value</th>
<th>Unit</th>
<th>Package</th>
<th>Manufacturer</th>
<th>Digi-key Part number</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2_R</td>
<td>1.8</td>
<td>pF</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>490-1378-1-ND</td>
</tr>
<tr>
<td>C3_R</td>
<td>6.8</td>
<td>pF</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>490-3563-1-ND</td>
</tr>
<tr>
<td>C5_R, C10_R</td>
<td>4.7</td>
<td>uF</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>490-3297-1-ND</td>
</tr>
<tr>
<td>C7_R</td>
<td>100</td>
<td>pF</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>490-1427-1-ND</td>
</tr>
<tr>
<td>C8_R</td>
<td>0.47</td>
<td>uF</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>490-3291-2-ND</td>
</tr>
<tr>
<td>C9_R</td>
<td>100</td>
<td>nF</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>490-1519-1-ND</td>
</tr>
<tr>
<td>D1</td>
<td>5.1V</td>
<td>Zener Diode</td>
<td>DO-214AC</td>
<td>Micro Commercial Co</td>
<td>SMAZ5V1-TPMSCT-ND</td>
</tr>
<tr>
<td>Decoder</td>
<td>HCS515</td>
<td>Decoder</td>
<td>14-SOIC</td>
<td>Microchip Technology</td>
<td>HCS515-I/SI-ND</td>
</tr>
<tr>
<td>IC2</td>
<td>ATTINY45</td>
<td>N/A</td>
<td>8-DIP</td>
<td>Atmel</td>
<td>ATTINY45-20PU-ND</td>
</tr>
<tr>
<td>L3_R</td>
<td>51</td>
<td>nH</td>
<td>_0603</td>
<td>Taiyo Yuden</td>
<td>587-2057-1-ND</td>
</tr>
<tr>
<td>L5_R</td>
<td>39</td>
<td>nH</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>490-1176-1-ND</td>
</tr>
<tr>
<td>L6_R, L7_R, L9_R</td>
<td>Ferrite bead</td>
<td>N/A</td>
<td>_0603</td>
<td>muRata Electronics</td>
<td>240-2377-1-ND</td>
</tr>
<tr>
<td>Q3</td>
<td>9.7941MHz</td>
<td>Oscillator</td>
<td>HC49</td>
<td>Abracon Corporation</td>
<td>535-9694-1-ND</td>
</tr>
<tr>
<td>Q4 - MOSFET</td>
<td>N-Channel</td>
<td>N/A</td>
<td>SOT 23-3</td>
<td>STMicroelectronics</td>
<td>497-3111-1-ND</td>
</tr>
<tr>
<td>R2</td>
<td>470</td>
<td>Ohm</td>
<td>_2010</td>
<td>Stackpole Electronics Inc</td>
<td>RMCF1/24705%RDKR-ND</td>
</tr>
<tr>
<td>R4_R, R6, PULLUP</td>
<td>100</td>
<td>kOhm</td>
<td>_0603</td>
<td>Vishay/Dale</td>
<td>541-100KHCT-ND</td>
</tr>
<tr>
<td>U1</td>
<td>MIRCF010</td>
<td>Receiver</td>
<td>8-SOIC</td>
<td>Micrel Inc</td>
<td>576-1962-5-ND</td>
</tr>
</tbody>
</table>

Table 4.

Parts List for the Receiver side
APPENDIX B

Figure 8

Transmitter Circuit Schematic
Figure 9

Receiver Circuit Schematic
Figure 10

Final Board Layout (transmitting side)
APPENDIX D

#include <avr/io.h>
#include <util/delay.h>
#include <inttypes.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#define F_OSC 1843200 //Clock Speed
#define baud 9600 //Baud Rate
#define baudnum F_OSC/16/baud-1 //baudnum calculation
#define keyport PORTD //Keypad Port
#define keyportddr DDRD //Data Direction Register
#define keyportpin PIND //Keypad Port Pins
#define col1 PD2 //Column1 PD2
#define col2 PD3 //Column2 PD3
#define col3 PD4 //Column3 PD4
#define TRUE 1
#define FALSE 0
#define RETRY 2
#define EXIT 3
#define LOCK 0
#define UNLOCK 1
#define EX 2
#define OK 3

//void led_on(int);
int getkey(void); //get input from keypad and buttons
void setulock(void); //used to set a new user code
void store_code(void); //stores the new user code
void keypad_init(void); //initializes all the ports
void delayus(unsigned char);
void delayms(unsigned char);
void send_bytes(char *, int);
void init_usart(unsigned int); //initializes the USART serial communication
unsigned char usart_receive(void);
void usart_transmit(unsigned char);
unsigned char translate(unsigned char);//translates the key inputs to values
int check(unsigned char *,unsigned char *,unsigned char);
void reset_receive(void);//resets the receive array after receiving a string
void get_scan(void);//decodes received string
void transmit_signal(void);//turns on all transmitter parts and send data pulse

unsigned char input[10], userlock[5], defaultulock[5], masterlock[10];

char enroll[15] = {0x41,0x01,0x00,0x05,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}; //enroll
cchar del[15] = {0x41,0x01,0x00,0x17,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}; //delete
cchar identify[15] = {0x41,0x01,0x00,0x11,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}; //identify
cchar en_no_scan[15] = {0x41,0x01,0x00,0x05,0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x6C,0xB5,0x0A}; //set unique code
cchar no_snan[15] = {0x41,0x01,0x00,0x11,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x6C,0xBF,0x0A};
volatile char receive[29];
cchar EEMEM storEEPROM[15]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

char hex = 0xB5; //Hex values to decode the received signal from Suprema
char hex1 = 0xBC;
cchar hex2 = 0xBE;
cchar hex3 = 0xB6;
cchar hex4 = 0x6C;
cchar hex5 = 0xBF;
cchar hex6 = 0x05;
cchar hex7 = 0xB6;
cchar hex8 = 0x11;
cchar hex9 = 0x6B;
cchar hex10 = 0xB9;
cchar hex11 = 0x61;
cchar hex12 = 0x41;

int lockstatus, keystatus=FALSE, newlock=FALSE, keyval; //declares initializes int global variables
unsigned char status; //declares unsigned char global variable
volatile int z=0; //declares z volatile int, z is counter in the receive array

void main()
{
    init_usart(baudnum); //init USAR
    UCSRB |= (1<<RXCIE0); //receive interrupt
    sei(); //global interrupts
    keypad_init(); //Port initialize

    while(1)
    {
        status = getinput(5); //waiting for input
        if(check(input,"12345",5)) //set unique code
        {
            setulock();
goto done;
        }
        if(status == TRUE)
        {
            if(newlock)
            {
                lockstatus = check(input,userlock,5);
            }
            else
                lockstatus = check(input,defaultulock,5); //default lock
        }
    }
}

//Rest of the code...
if(lockstatus)
{
    PORTC |= 0x01; //lock deactivated
    _delay_ms(250);
    while(1)
    {
        leave0;
        keyval=getkey();
        switch (keyval)
        {
            case 1: //enroll
            while(1)
            {
                keyval=getkey()
                switch(keyval)
                {
                    case 11:
                    goto leave0;
                    case 12:
                    PORTD |= 0x20;
                    _delay_ms(1000);
                    _delay_ms(1000);
                    send_bytes(enroll,15);
                    _delay_ms(250);
                    PORTC |= 0x02;
                    _delay_ms(250);
                    goto leave0;
                }
            }
        }
        case 2: //delete
        while(1)
        {
            keyval=getkey();
            switch(keyval)
            {
                case 11:
                goto leave0;
                case 12:
                PORTD |= 0x20;
                _delay_ms(1000);
                _delay_ms(1000);
                send_bytes(del,15);
                _delay_ms(250);
                goto leave0;
            }
        }
        case 11: //cancel
        PORTC &= ~0x01;
        goto leave;
    }
}
leave;
}
else
{
    PORTC |= 0x04;
    _delay_ms(250);
    PORTC &= ~0x04;
/** USART COMMUNICATION MODULE

SYNTAX:   void init_usart(unsigned int ubrr)
DESCRIPTION:   Initialize USART COMMUNICATION
PARAMETER:   NONE
RETURN VALUE:  USIGNED INT
NOTES:

void init_usart(unsigned int ubrr)
{
    UBR0H = (unsigned char)(ubrr>>8);  // set baud rate
    UBR0L = (unsigned char)ubrr;      // set baud rate
    UCS0 = (1 << RXEN0) | (1 << TXEN0);  // enable RX, TX and RX interrupt
    UCS0C = (0 << USBS0)| (3 << UCSZ00);  // asynchronous, 8 data, 2 stop
}

SYNTAX:   void send_bytes(char *s, int length)
DESCRIPTION:   Sends data strings
PARAMETER:   none
RETURN VALUE:  character string, integer
NOTES:   This function calls usart_transmit() to transmit data

void send_bytes(char *s, int length)
{
    reset_receive();
    while (length--)
    {
        usart_transmit(*s);
        s++;
    }
}

SYNTAX:   void usart_transmit(unsigned char c)
DESCRIPTION:   Transmits data bytes
PARAMETER:   none
RETURN VALUE:  character
NOTES:   This function transmits single bytes

void usart_transmit(unsigned char c)
{
    while(!(UCSR0A & (1 << UDRE0)));  // wait until UDR ready
    UDR0 = c;                     // send character
}

SYNTAX:   unsigned char usart_receive(void)
DESCRIPTION:   reads a character
PARAMETER:   none

unsigned char usart_receive();
unsigned char usart_receive(void)
{
    while (!(UCSR0A & (1<<RXC0)));    //wait for data to be received
    return UDR0;                      //return data from buffer
}

ISR(USART_RX_vect)
{
    cli();                             //disable interrupts
    while (!(UCSR0A & (1<<RXC0)));     //wait for data to be received
    receive[z]=UDR0;                   //put input in receive[] array
    z++;                               //increment z
    sei();                             //enable interrupts
}

int check(unsigned char *first, unsigned char *second, unsigned char len)
{
    unsigned char i=0;                  //checks to see if input matches default change password
    for(i=0;i<len;i++)
    {
        if(first[i]!=second[i])
            return FALSE;
    }
    return TRUE;
}

void setulock()
{
    PORTC &= ~0x01;                    //puts the user to set their own user lock
}
_delay_ms(250);
PORTC |= 0x01;
char status;
retry:
status = getinput(10);
if(status == TRUE)
{
    if(check(input,masterlock,10))
    {
        PORTC &= ~0x01;
        _delay_ms(250);
        PORTC |= 0x01;
        retry1:
        status = getinput(5);
        if(status == TRUE)
        {
            newlock = TRUE;
            store_code();
            _delay_ms(250);
            goto exit;
        }
        else if(status == RETRY)
        goto retry1;
        else if(status == EXIT)
        goto exit;
    }
    else
    {
        goto exit;
    }
}
else if(status == RETRY)
    goto retry;
else if(status == EXIT)
    goto exit;
exit:;
}

/**
 * getinput(unsigned char max)
 * Allows the user to set their own user lock
 * Parameter: unsigned character
 * Return Value: character
 * Notes:
 */
char getinput(unsigned char max)
{
    unsigned char i,key;
    //takes key presses and assigns them to input
    i=0;
    while(1)
    {
        while(!((key=getkey())));
        key = translate(key);
        input[i]=key;
        if(key=='e') //Enter button pressed
        {
            return TRUE;
        }
    }
}
else if(key=='i')
{
    PORTD |= 0x20;
    _delay_ms(1000);
    _delay_ms(1000);
    send_bytes(identify,15);
    _delay_ms(250);
    PORTC |= 0x02;
    _delay_ms(250);
}
else
{
    i++;
    if(i>max)
    {
        _delay_ms(250);
        return RETRY;
    }
}
}

/***************************************************************************/
*****
SYNTAX:
void store_code()
DESCRIPTION: Stores new user codes
PARAMETER: none
RETURN VALUE: none
NOTES: **************************************************************/
void store_code()
{
    unsigned char i;
    for(i=0;i<5;i++)
    {
        userlock[i]=input[i];
        PORTC &= ~0x01;
        _delay_ms(250);
        PORTC |= 0x01;
    }
}

/***************************************************************************/
* KEYPAD FUNCTION MODULE
******************************************************************************/
/*******************************************************************************/
SYNTAX:    void init_ports()
DESCRIPTION: Initializes ports for correct direction and initial value
PARAMETER: none
RETURN VALUE: none
NOTES: **************************************************************/
void keypad_init(void)
{
    keyportddr = 0x20;  //Pin 5 output, the rest are inputs
    keyport = 0xDF;     //Pin 5 low, the rest are high initially
    DDRB = 0x3E;        //Pins 1-5 inputs, the rest are outputs
    PORTB = 0x00;       //All pins low initially
    DDRC = 0x37;        //Pins 0,1,2,4,5 are outputs, the rest are inputs
    PORTC = 0x00;       //All pins initially low
/**
 * SYNTAX: void store_code()
 * DESCRIPTION: Stores new user codes
 * PARAMETER: none
 * RETURN VALUE: none
 * NOTES: */

int getkey()
{
    if(receive[0]==hex12) { //Check if data has been received
        get_scan(); //Call decode_receive if data has been received
        int i,key=1; //initially sets key = 1
        for(i=0;i<4;i++) //Loop for 4 rows
        {
            PORTB &=~(0x02<<i); //Make rows low one by one
            _delay_ms(5);
            if(!(keyportpin & (1<<col1))) //check COL1
            {
                while(!(keyportpin & (1<<col1))); //wait for release
                return key; //return pressed key value
            }
            if(!(keyportpin & (1<<col2))) //Check COL2
            {
                key += 1; //Second key pressed
                while(!(keyportpin & (1<<col2))); //wait for release
                return key; //return key value
            }
            if(!(keyportpin & (1<<col3))) //Check COL3
            {
                key += 2; //Third key pressed
                while(!(keyportpin & (1<<col3))); //Wait for release
                return key; //Return value
            }
            key +=3; //increment key by 3 for next row
            PORTB |= 0x02<<i; //make read row high again
        }
        return 0; //return false if no key pressed
    }
}

/**
 * SYNTAX: unsigned char translate(unsigned char keyval)
 * DESCRIPTION: Translates key inputs
 * PARAMETER: unsigned char
 * RETURN VALUE: unsigned char
 * NOTES: This translates the key values to the correct number or letter */

unsigned char translate(unsigned char keyval)
{
    if(keyval<10) //If key value < 10 remains the same
    {
        return keyval+'0';
    }
    else if(keyval==10) //If key = 10 set to char 'i'
    {
        return 'i';
    }
    else if(keyval==11) //If key = 11 set to char 'c'
    {
        return 'c';
    }
    else if(keyval==12) //If key = 12 set to char 'e'
    {
        return 'e';
    }
void reset_receive(void)
{
    int x=0; //initialize counter x
    z=0; //reset global counter z
    PORTC &= ~0x02; //turn off yellow LED from enroll/identify
    PORTB &= ~0x20; //turn off power to suprema board
    while(x<30)
    {
        receive[x] = 0x00; //loop writing 0x00 to full receive array
        x++; //increment counter
    }
}

void get_scan(void)
{
    {
        PORTC |= 0x04;
        _delay_ms(1000);
        PORTC &= ~0x04;
        _delay_ms(1000);
        PORTC |= 0x04;
        _delay_ms(1000);
        PORTC &= ~0x04;
        reset_receive();
    }
    {
        transmit_signal();
        _delay_ms(250);
        PORTC |= 0x01;
        _delay_ms(1000);
        PORTC &= ~0x01;
        _delay_ms(1000);
        PORTC |= 0x01;
        _delay_ms(1000);
        PORTC &= ~0x01;
        reset_receive();
    }
{ 
    PORTC |= 0x01;
    _delay_ms(1000);
    PORTC &= ~0x01;
    _delay_ms(1000);
    PORTC |= 0x03;
    _delay_ms(1000);
    PORTC &= ~0x03;
    reset_receive();
}

{
    PORTC |= 0x04;
    _delay_ms(1000);
    PORTC &= ~0x04;
    reset_receive();
}

if((receive[27]==hex9) || (receive[28]==hex10))  //enroll bad scan
{
    PORTC |= 0x04;
    _delay_ms(1000);
    PORTC &= ~0x04;
    reset_receive();
}

if(receive[27]==hex11)  //enroll good scan
{
    PORTC |= 0x01;
    _delay_ms(1000);
    PORTC &= ~0x01;
    _delay_ms(1000);
    PORTC |= 0x01;
    _delay_ms(1000);
    PORTC &= ~0x01;
    _delay_ms(1000);
    PORTC &= ~0x01;
    reset_receive();
}

} //******************************************************************
SYNTAX:    void transmit_signal()
DESCRIPTION:  Transmits signal on good identify
PARAMETER:  none
RETURN VALUE: none
NOTES:      This function turns on all the transmitter circuitry as well as sends the signal to the encoder.

void transmit_signal()
{
    PORTC |= 0x20;    //Turn on PC5 (power for transmitter)
    _delay_ms(1000);
    PORTC |= 0x10;    //Pull PC4 (S0 data line) high on encoder to send signal
    _delay_ms(500);
    PORTC &= ~0x30;   //Pull PC4 and PC5 low to stop transmission and power off
    _delay_ms(250);
}
APPENDIX E

/*********************************************
* Team ThumbStart       University of Idaho
* Chip type           : ATtiny45
* Code Date: 12/07/08
* PB0 input from S0
* PB2 output to relay
*********************************************/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

void main()
{
    DDRB = 0x04;               // set PB2 output and PB0 for input
    PORTB &= ~0x04;            // set PB2 low initially
    while (1)                  // loop forever
    {
        if((PINB & 0x01) == 1) // constantly check if PB0 gets pulled high
        {
            PORTB |= 0x04;     // Turn on output for relay
            _delay_ms(1000);   // Stay in loop to keep relay on
            while(1);
        }
    }
}
Important Data Sheets

ATMEGA48

http://www.atmel.com/dyn/resources/prod_documents/2545S.pdf

ATTINY45


Suprema SFM3010-FC


HCS300 Encoder


HCS515 Decoder


MICRF113 Transmitter

http://www.micrel.com/_PDF/micrf113.pdf

MICRF010 Receiver

http://www.micrel.com/_PDF/micrf010.pdf