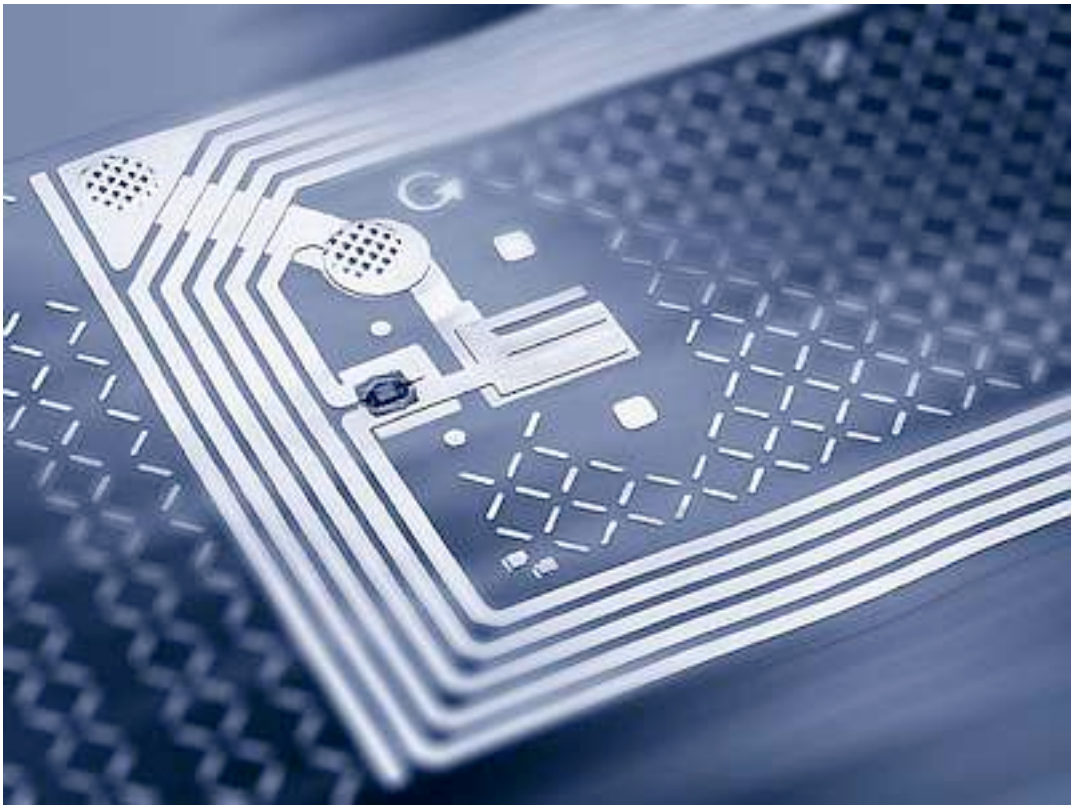
Continuous Polling RFID System



Ian Czop
Eric Nord
Alexander Tu

Advisor: A. Lee Swindlehurst

Abstract

Radio-Frequency Identification (RFID) is a popular technology used in many industries as a means of identification. Many countries, including the United States now embed their passports with RFID tags. RFID tags are cheap, reliable, and many are passive so they don't require any batteries to respond to the request of an RFID reader. One particular interesting use of RFID systems is by automobile companies to add to user convenience when getting into and starting a vehicle.

As a result, our interests were immediately sparked when this project was proposed to us by **RFID Options**, through their Patent Lawyer, Donald Bollella. They introduced us to their patent on Continuously Polling RFID technology and sponsored our project. They asked us to design a working model of their system as a proof of concept.

Based on their input, we set the goal of our project to be the development of an RFID Anti-Carjacking and Child Monitoring system. The system has two main purposes: First, prevention of engine ignition without the proper RFID Driver's tag and the prevention of normal car operation should the Driver's tag suddenly go out of range (ie. a carjacking situation), and second creation of a way of alerting a parent if a child has been left unattended in a car. In order to achieve these goals, both the driver and child will have an RFID chip on their person. The driver will have a wireless key fob, henceforth referred to as the Driver's tag and the child will have some article of clothing or accessory embedded with an RFID chip (e.g. a bracelet).

Table of Contents

Issues with Current Vehicular Single Polling RFID Systems

Carjackings are an unfortunate reality in today's society. And while crime cannot be completely stopped, it can be slowed and discouraged with better automobile security technology.

Figure 1: A Carjacking in Action

Current RFID car systems detect the presence of a wireless RFID key fob and perform such actions as automatically unlocking car doors and allowing engine ignition with the press of a button. However, the issue with these systems is that during the operation of a vehicle, the presence of the key is only checked once. If a driver forgets to turn the engine off and walks always, the car will continue running until it runs out of gas or the owner comes back and shuts it off. Even worse, if a driver is carjacked while idling at a stop sign or traffic light, the criminal can drive away without possessing the actual physical key fob.

The solution to this problem is a system that implements Continuous Polling, that is, a system that systematically checks for the presence of a tag. When the engine is on,

but the key fob is not in range, the engine will issue a warning and if the key is not reintroduced within a certain period of time, the gas to the engine will be cut off and an intruder will be unable to get away with the vehicle.

The Child Safety Issue

Parents can sometimes be careless and accidentally leave their young children unattended in a vehicle. Children have a lower tolerance for extreme temperatures and are very easily susceptible to heat stroke and hyperthermia. In fact, a study done by San Francisco State University (http://ggweather.com/heat/) found that 361 children have died of hyperthermia between 1998 and 2007 due to being left unattended in a vehicle. A CDC published estimate, **Table 1**, shows that between July 2000 and June 2001, over 9000 children were injured so badly by being left unattended that they required hospital treatment.

TABLE 1. Estimated number and rate* of injuries treated in hospital emergency departments among children aged ≤14 years who were left unattended in or around motor vehicles — United States, July 2000–June 2001

Characteristic	No.	%	Rate	(95% CI†)
Age (yrs)				
0– 4	3,800	41.5	20.1	(8.5–31.7)
5–14	5,360	58.5	13.5	(8.4–18.5)
Sex				
Male	5,674	61.9	18.9	(11.0–26.8)
Female	3,486	38.1	12.2	(6.1–18.2)
Total	**9,160**	**100.0**	**15.6**	**(9.1–22.1)**

* Per 100,000 population.
† Confidence interval.

Table 1: Injuries to Children Left Unattended in or Around Motor Vehicles

These problems could be solved easily with Continuous Polling by checking for the presence of a child RFID tag and the absence of the driver's key fob. The vehicle can then alert the negligent parent by sounding an alarm and protect the child by cracking the windows slightly in high temperature situations.

Implementation

Our goal was to create a prototype of a Continuous Polling RFID System. We decided to split the system into two main parts: an RFID reader that continuously polls for the presence of tags and a microcontroller that will simulate the Engine Control Unit and react accordingly to the combination of tags missing/present. To better show the relationship between these two parts, a block diagram is presented below in **Figure 2**.

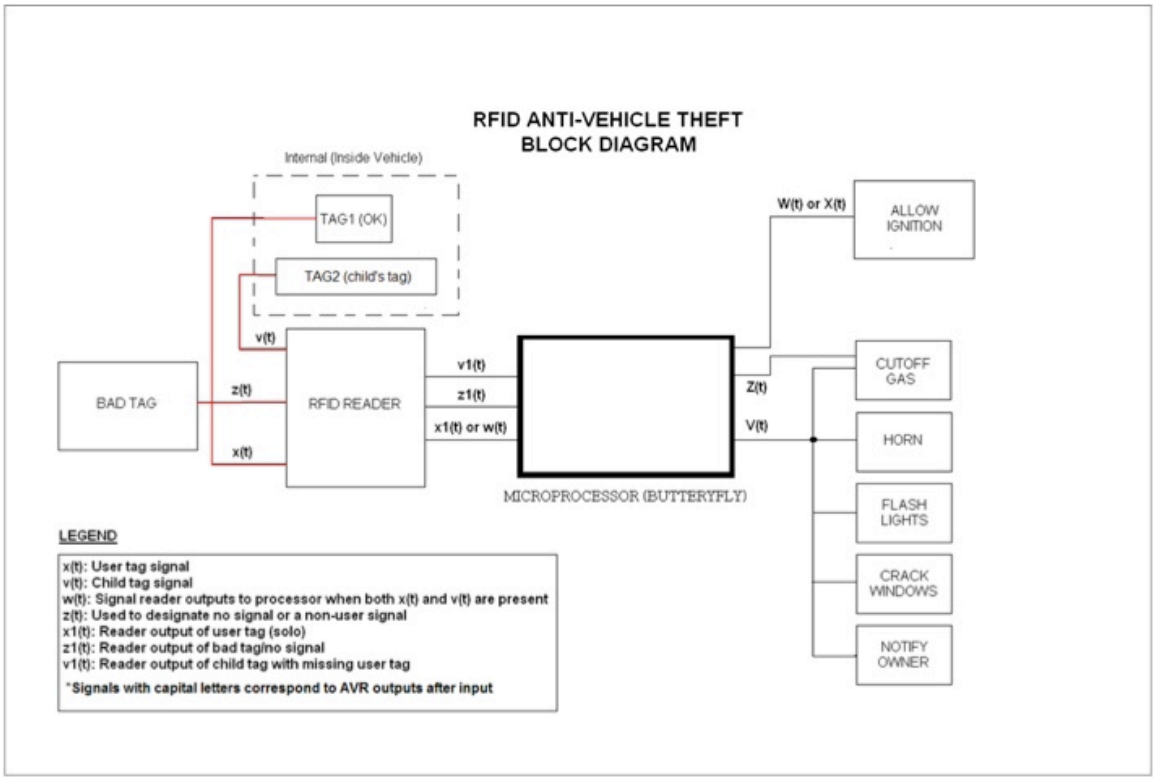


Figure 2: Block Diagram Demonstrating Connections

A flowchart is also presented in **Figure 3** to further clarify the behavior of the system. To begin, the engine status is checked and if the engine is off, the system will check to see if the child is left unattended and react accordingly. If no child is present,

then the system will check to see if the ignition button is pressed and will start the engine of the Driver's tag is in range.

While the engine is on, the system will continually check for the presence of the Driver's tag every fifteen seconds. This loop will run as long as the engine is on. However, if the Driver's tag is not present, the system will issue a warning and allow fifteen seconds for the Driver's tag to be reintroduced. If that fails, the engine will be shut off.

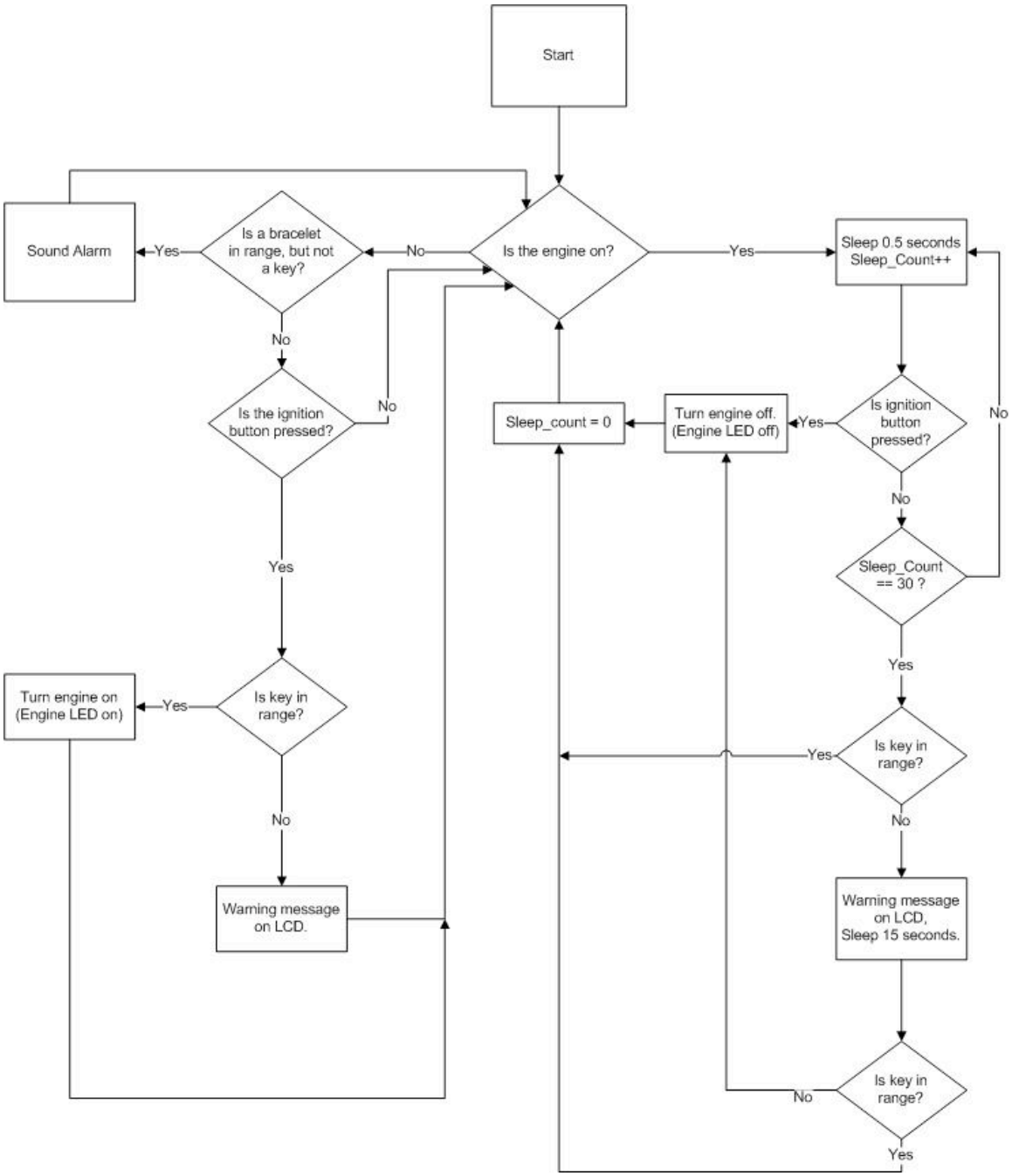


Figure 3: Flowchart of the Continuous Polling RFID System

Atmel AVR Butterfly

As purchasing an Engine Control Unit was not feasible for this project, we utilized Atmel's AVR Butterfly to emulate the Continuous Polling system and its reactions.

The Atmel AVR Butterfly is an evaluation board that uses the ATmega169 8-bit RISC microprocessor. We chose this particular module because it contains an LCD display for easier debugging of programs as well as for its bootloader firmware which allows the device to be flashed thru UART. This reduced costs, as a simple three wire UART programmer can be made for about $7, while an ISP programmer costs upwards of $30.

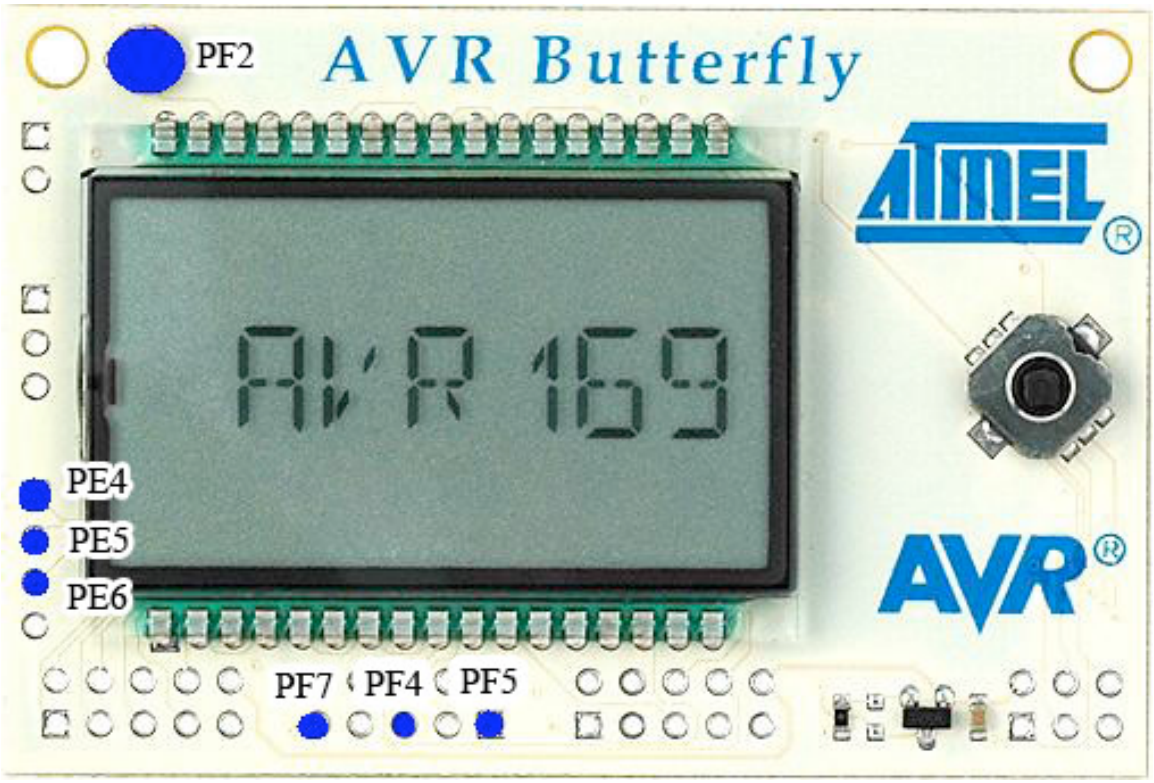


Figure 4: Input and Output Ports Utilized on the AVR Butterfly

We wrote the code and compiled it in Programmers Notepad, a GNU GCC compiler included with the open source software development package WinAVR. Thanks to the bootloader, we were able to use AVRPROG included with AVR Studio to flash our code onto the Butterfly. For the sake of brevity, the code is separately attached. Please look at **main.c** to see the code we used in the Butterfly. It will add insight into the implementation of this project.

SkyeTek M9 RFID Reader

To choose which RFID reader would be best for our project there were several criteria that we needed to fill such as:

1. The reader had to be relatively small so that once it was installed in the car the owner would not even know it was there.
2. The reader had to have a range that could reliably read tags positioned anywhere in a moderately sized vehicle.
3. The reader had to have an interface that was compatible with the microprocessor that we chose.
4. The reader would preferably be compatible with a large number of tags.

Once we started searching for reader that fit all of these necessary criteria we slowly noticed how limited the consumer market was in this area. The largest problem was that the readers that were available had a very limited range, which was in the centimeter range, whereas we needed a reader with a range of at least two meters. Also, many of the readers on the market were only compatible with their own proprietary tags, which was not ideal for our application. If the reader was compatible with third party tags then manufactures would have many more tag options if they wanted to pursue this system. After taking all of these requirements into consideration we decided to choose the SkyeTek M9 RFID reader, **Figure 5**.

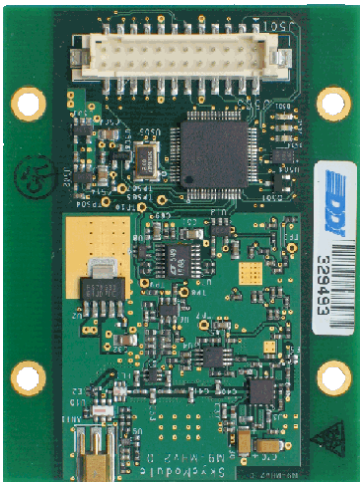


Figure 5: SkyeTek M9 reader

The M9 reader is one of the few RFID readers on the market that adequately met or surpassed the above requirements. Some of the desirable aspects of the M9 reader are:

1. The M9 is an extremely small reader; it is about the size of a business card. The reader's antenna is obviously a lot bigger than the actual reader though, but it would probably be possible to integrate the antenna into a cars body panels, roof or floor so that it would not take up too much space.

2. The M9 has a range over 3.5 meters with the proper antenna, which is more than enough coverage to read a tag placed anywhere in a large vehicle.

3. The M9 has many different types of interfaces available to communicate to microcontrollers and windows based computers. These include: USB, TTL, RS-232 (when using a host interface board), I2C, and SPI.

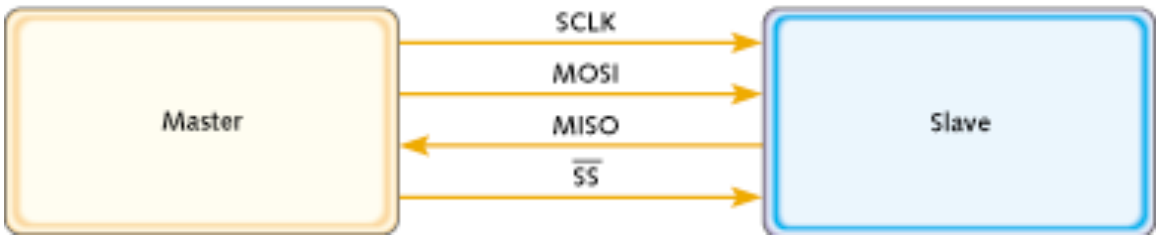
4. The M9 is also compatible with a wide range of third party tags, which is a huge plus. Because it is compatible with so many tags it might be possible for companies to upgrade their current RFID systems to more advanced continuous polling systems, without having to purchasing new tags.

Another criterion that makes the M9 reader a good fit for the project is the fact that it operates in the ultra high frequency (UHF) range. UHF goes from 300 MHz to 3GHz and the benefit of operating in this range is that the waves are physically shorter so the antennas can be smaller, while still giving an adequate RFID read range. One downside of operating in this range though is that the waves have trouble passing through moisture, which could considerably shorten the read range in rainy weather. The waves also are more susceptible to noise, but since the M9 RFID reader has a range of over 3.5 meters, the noise and the shortening of the read range in rainy weather shouldn't affect the system too drastically.

Choice of Interface

As stated before, the M9 RFID reader has multiple interface choices that can be used to communicate to microcontrollers or windows based computers such as USB, TTL, RS-232 (when using a host interface board), I2C, and SPI. Although, there are several different interfaces, there are only two that are compatible with the Butterfly, which are SPI and RS-232. We chose to use the SPI interface since it is a fairly simple interface to understand conceptually and it is also an extremely reliable interface. RS-232 in contrast is more complex since you have to with deal issues such as baud rates.

SPI stands for serial peripheral interface and it communicates using four different lines: SCLK or serial clock, MOSI or master out slave in, MISO or master in slave out, and SS, which is the slave select line.



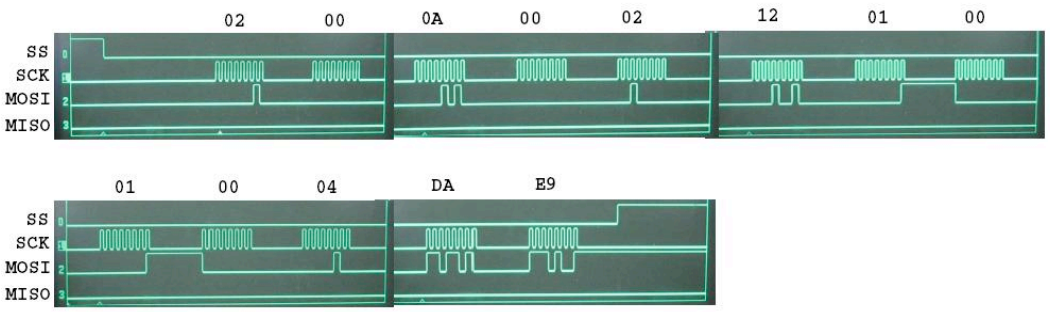
In SPI each component must be considered a master or a slave. There can only be once master, but there may be multiple slaves. In this case the Butterfly is the master and the RFID reader is the slave. The way SPI works is that when the master wants to send data it will turn the SS line from 1 to 0, send a clock signal over the SCLK line, and then send the data over the MOSI line. Once all the data has been sent then the master

Figure 6: SPI interface overview

changes the SS line from 0 back to 1. When the master wants to receive data from the slave then it is basically the same operation, except instead of sending data over the MOSI line the master sends a dummy bit or (0x00). When the master is sending the dummy bit over the MOSI line the slave actually sends its data over the MISO line in sync with the clock singal that the master sends over the SCLK line. A sample waveform as viewed from a logic analyzer can be seen in **Figure 7**.

Another thing important to note is that unlike sending data from master to slave where you wait until the end of the send string to pull the SS line from 0 back to 1, when the master is receiving data it must pull the SS line from 0 to 1 after every 8-bit character. This is not usually the case, but the reader (slave) is set up so that you must do this so that the reader knows that the Butterfly (master) has received the 8-bit character.

Request: Host -> M9
(Read Firmware Version - 02 00 0A 00 02 12 01 00 01 00 04 DA E9)



Response: M9 -> Host
(First 3 Bytes - 02 00 0A 12 01 00 04 01 00 01 7A DF 5C)

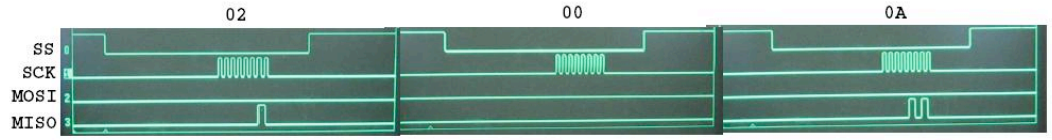


Figure 7: Sample SPI signal provided by SkyeTek

Complications

As with any engineering project during the course we encountered several problems. One problem we encountered during the beginning of the project was with flashing the AVR Butterfly. In fact, during our programming attempts, the bootloader was somehow able to brick (that is, corrupt the firmware) of two of our Butterflies. As the UART programming only works through the bootloader, we could not use UART to reflash the firmware. We ended up having to buy an ISP Programmer off eBay to restore functionality.

Another problem that we encountered during the project was getting the transistor, which we designed to control the flow of current to the RC car controller, to turn on and shut off. Our design was fairly simple – the Butterfly would be connected to the gate of an NMOS transistor and the drain and the source would be connected to the 9V battery and the controller respectively. When we wanted the car to run the Butterfly would apply a voltage to the gate and allow current to flow and when we wanted to disable the car we would have the Butterfly ground the gate. The problem occurred when we tried to turn on the transistor. For some reason after we applied voltage to the gate of the NMOS transistor, the NMOS would not turn on until we floated the node for several seconds. To solve this problem we simplified the circuit by removing the 9V battery and the transistor and instead had the Butterfly power the RC car controller. At first we thought this might not work since the Butterfly only has a maximum output of 5V, but after testing it proved successful, which simplified the circuit and actually lowered the final cost without removing any functionality.

The last problem that we encountered dealt with the interface that we decided to use to communicate between the reader and the Butterfly. For some reason they were having trouble communicating properly through SPI so we decided to talk to Professor Jenks, who had previous experience with the interface. He was able to give us valuable information on how SPI communicated and also sent us some sample code. Unfortunately the M9 reader did not communicate through SPI like most standard components did, which made communicating through SPI different and more difficult. At this point though, after several weeks of working with SPI, we decided that instead of pursuing the troublesome interface we would instead focus on demonstrating the proof of concept of the project. We did this by creating a hardware model that demonstrated all of the features of a continuous polling RFID system.

Hardware Model

In order to demonstrate all the functionality of an RFID continuous polling system used in an automobile we decided to create a hardware model. Instead of getting the tag information directly from the RFID reader we connected switches on the microcontroller's ports, where one represented the Bracelet and the other represented the Driver's tag. When a switch was flipped it connected the Butterfly's respective active low port to ground, which told the Butterfly that the tag was in range. A model RC car was also implemented in order to show a "real life" demonstration of how power to the engine would be cut off if the driver's tag was not present. The hardware model's circuit can be seen below in **Figure 8**.

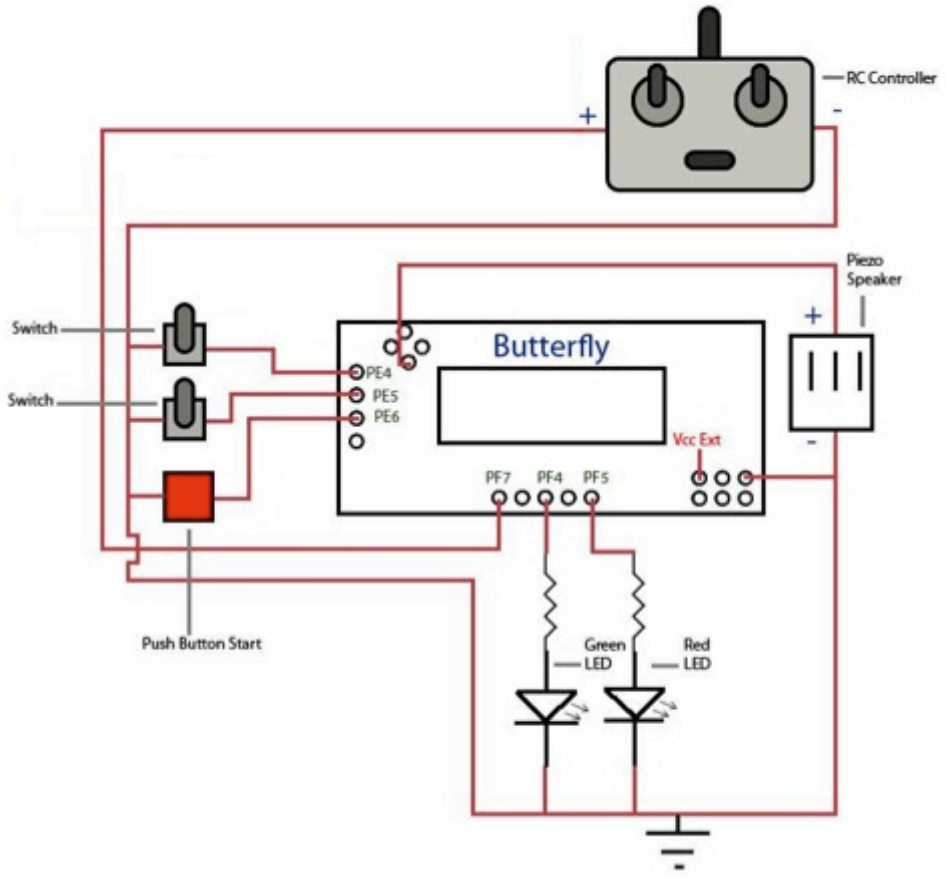


Figure 8: The hardware model of system

The green LED represents the Engine On state, which can only be achieved if the Driver's tag is in range and the ignition button is pressed. This case is possible when the top switch completes the connection between ground and port PE4 on the Butterfly (ON position). The red LED and the piezo speaker are activated when the Bracelet is left alone in the car. This case is possible when the top switch is in the off position and the bottom switch completes the connection between ground and port PE5 (ON position).

In the Engine On state, the Butterfly, in addition to turning on the green LED, also applies +5V through port PF7, which turns on the RC controller and allows the RC car to run. When the ignition button is pressed again or the driver's tag moves out of range (the top switch is in the off position) then the Butterfly grounds port PF7, which cuts the voltage to the RC controller thus shutting off the RC car. This system effectively demonstrates all the functionality that a continuous polling RFID system would have.

To show how continuous polling could be beneficial as an anti-carjacking device, we created a Windows-based program that used continuous polling to turn on the car and shut it off when the driver's tag is in range, this program can be viewed in **sloop.cpp**, which is attached. This program is based on a small section of code that was provided in the developer's kit that came with the RFID reader. The code was then edited and adapted to create a program that displays messages that would presumably be displayed on the owner's dashboard if this system were installed in a car.

To use this program you connect the SkyeTek M9 RFID reader to a Windows based computer through USB. When the program is run it goes into a while loop that continuously checks for tags and displays a message whenever the driver's tag moves sufficiently away from the reader's antenna or comes into range. If the driver's tag is gone for an extended period of time it will display a warning message that indicates the car has been shut off.

Alternative Options

One thing we could have done differently would be to substitute the SPI connection with RS-232 or a different interface. RS-232 is a more popular interface and because of that there is a lot more information available on it. One of the main problems we had with the SPI was the lack of available (especially introductory) materials since it was the first time we were all exposed to it. However RS-232 is more difficult so this would be one tradeoff; more information, more difficulty

Another alternative would have been to choose a different RFID reader system. The main reason for choosing the SkyeTek reader was that advertised range of ~3.5m (radius). This was perfect for a real life application of our project. However, the reader would only acknowledge a tag when it was within a few centimeters, and couldn't distinguish between tags properly in some cases. Also, sections of the sample C code provided with the kit did not work properly when implemented exactly as given. Either the reader was malfunctioning or the code samples were flawed. A simpler reader could have proved easer to interface, but would have had even less range and less tag options.

One further option we could have explored is an alternative microcontroller like a PIC. Also, it would have been useful to choose a microcontroller that implemented a USB or I2C interface so we could have had different interface options.

Item	Quantity	Cost	Total Cost
SkyeTek RFID Developer's Kit	1	$1000	$1000
Atmel AVR Butterfly	3	$21	$63
ISP Programmer	1	$28	$28
Breadboard	1	$20	$20
Protoboard	1	$5	$5
Batteries (9V, 4 AA)	2	$15	$15
RC Car	1	$20	$20
9V Battery Cap	1	$2	$2
Project Display Box	1	$6	$6
D-Sub 9 Connector	3	$3	$9
D-Sub 9 Hood	3	$3	$9
Misc. Wires	1	$5	$5
Serial to USB adapter	1	$8	$8
Switches	3	$2	$6
Electrical Tape	1	$3	$3
LED	2	$2	$4
TOTAL COST			**$1203.00**

Cost and Time Analysis

The cost of each item as well as the quantity and total cost are documented below in **Table 2**. Not including the RFID Developer's Kit, the cost of items stayed around $200. We anticipated spending about $100 each so we overestimated by about 30%.

Table 2: Itemized Description and Cost

Task	Approximate Time(hrs)
Preliminary Planning/Layout/Meeting	30
Working on SPI	20
Functionality code	30
Butterfly hardware/flashing	20
Presentation/Documentation	20
TOTAL APPROX. HOURS	120

Table 3: Time by Task and Total Hours

Concerning safety, nothing during the process of our project put our team in any serious risk. The soldering iron was the most hazardous piece of equipment in which one team member received a minor burn on his arm, but other then that it was harmless. Also, there wasn't any risk of electrocution save putting our fingers in an electrical socket which much to our delight did not happen. Proper precautions were also taken when drilling the project box.

Quality was assured by testing the hardware portion all along the way rather then just the final product. A multimeter was utilized to check for continuity in the circuit. Software quality was also assured by debugging and multiple tests through all possible combinations of inputs and their correctly observed outputs. Much time and effort was put into the final presentation of the hardware by placing the protoboard inside the control box and mounting the switches, LEDs, and Butterfly in an aesthetically pleasing manner as can be seen in **Figure 9**.

Figure 9: Control Box and RC Car

Feasibility

With any products, there are concerns of marketability. While it would be beneficial to implement this technology on all automobiles, RFID readers are expensive and most people would be willing to forego adding the feature to their vehicle in order to save money. With auto manufacturer price gouging, implementing an RFID system onto a car would probably cost upwards of $4,000. One solution for the cost would be to offer incentives for having a Continuous Polling system installed in a car such as tax breaks or discounts on car insurance. However, this cannot be guaranteed, so therefore the primary target for this feature would be more expensive vehicles that already have RFID systems implemented in them. Changing the system to work in a Continuous Polling manner would be simple to implement and a very beneficial deterrent to crime as well as child safety.

Though the system thinks about the safety of other drivers on the road by gradually decreasing fuel to the engine rather than just shutting it off instantly, it is still very dangerous to have a vehicle stop in the middle of a road. Should it happen to be in the middle of a freeway, it could endanger nearby drivers as well as the criminal. The ethics of having such a system is questionable and legal issues may arise out of it.

Conclusion

A Continuous Polling RFID System could solve many of the problems that the current Single Polling systems have. It would allow the ECU to shut the car off if the Driver's tag is out of range in such situations as a driver leaving his car running on accident or someone carjacking the owner's car. It would also allow the ECU to warn the driver that they have left their baby in the car unattended and could take further measures

to protect the baby in dangerous weather, e.g. cracking the windows when the temperature is above a certain threshold.

We did have some problems during this project, but senior projects are meant to challenge the individuals involved and teach them aspects of engineering that you don't necessarily learn in a classroom environment. During the beginning of the project most of the concepts involved were completely foreign to us. We had never programmed a microcontroller, communicated with a microcontroller, worked with RFID, or even heard of SPI. Throughout this project we learned and worked with all of these concepts and we created a working system that demonstrated some of the features a continuous polling RFID system could have. Although, we weren't able to successfully implement SPI, we believe that if given more time and access to a logic analyzer, we would have been able to.

One of the benefits of choosing such a daunting senior project is the fact that we got to work on a real patent that hadn't been implemented before and practically experience what it's like working in the industry. In fact since the company RFID Options sponsored us we were actually working with the industry. The broadness of the project also allowed us to deal with very different fields within electrical engineering, such as hardware design, microcontroller programming, interfacing, and radio frequency identification. This ultimately broadened our understanding of how these different fields come together and interact to create a final product. We gained a lot of useful experience during this project and it should prove to be extremely useful in our futures, whether it's pursing a graduate degree or working in the electrical engineering industry.

Works Cited

Cover picture: http://edumicator.org.uk/wp-content/uploads/2007/11/rfid-tag.jpg

Figure 1: http://www.tunersgroup.com/images/large/large76.jpg

Figure 4: http://www.atmel.ru/Disks/AVR%20Technical%20Library/tools/developer/Butterfly/pic/Butterfly_high.jpg

Figure 5: http://www.meshedsystems.com/91_images/Skyetek/M9%20top%20frei%20v01%20klein.gif

Figure 6: http://img.cmpnet.com/embedded/gifs/2002/0202/0202bcfig1.gif

Figure 7: E-mailed to us by SkyeTek.

Table 1: http://www.cdc.gov/MMWR/preview/mmwrhtml/figures/m126a3t1.gif