

# Car Jackers

# Project Report

CS 4710 - Fall 2012

## Team members

Jeremy Bonnell - [jjbonne@gmail.com](mailto:jjbonne@gmail.com)

Tong Wu - [tongwutwu1@gmail.com](mailto:tongwutwu1@gmail.com)



University of Utah

## Abstract

The Car Jacker is a device that allows a person to control settings of their car with an Android smartphone. Vehicle features controllable by the Car Jacker include the ignition, locks, and climate control settings. Whether one would like to remotely heat and defrost their car on an ice cold morning, or cool it down in the heat of the blistering sun, the Car Jacker will get the job done.

# I. Functional Description

The Car Jacker has 3 basic functions to control a vehicle via a wireless phone: remote start/stop of the vehicle, remote lock/unlock of the vehicle's doors, and control of the vehicle's heating/cooling system. The heating/cooling settings controllable by the Car Jacker are the fan speed, temperature setting, vent position, rear defrost, and air conditioner (AC). In addition to the basic functions, the Car Jacker has a safety feature to prevent a user from interacting with the system while driving, as well as a theft deterrent feature to prevent an idle car from being stolen while the owner is away.

To use the Car Jacker, a software application must be installed on an Android smartphone and hardware must be installed in the vehicle. To ensure they are paired correctly, the software is defined to a particular hardware device. The user can then launch the Android interface, adjust the settings, and press the "Send" button. A *text message* is sent by the phone to a cellular shield. The cellular shield is a device that enables text messaging services within the vehicle. The cellular shield then transmits the message to an Arduino microcontroller. The Arduino microcontroller is a small hardware device that runs a software application that is waiting for messages from the phone. The cellular shield and Arduino will be covered in greater detail in later sections. Once a valid message has been received, the Arduino sends signals to a circuit board to bypass the power supplies of the ignition switch, lock switches, and dials on the heating/cooling panel. The Arduino can then switch relays to direct the power to the actual motors - i.e. the starter motor, blower motor, etc... Once the settings have been enabled, the microcontroller sends a confirmation message back to the cellular shield which sends the message back to the phone in the form of a *text message*. The user can then select new settings or shut the Car Jacker system down.

The safety feature was developed to prevent users from operating the system while driving. The Car Jacker system only works if the vehicle shift knob is in the "Park" position. Once the driver places the key in the ignition and moves the shift knob out of "Park", the system is disabled and control will return to the dashboard settings. The microcontroller detects this occurrence and sends a message back to the phone. The system will not reinitialize until the shift knob has been placed back into the "Park" position.

The theft deterrent feature was developed to prevent the driving of the car without the key. The brake pedal must first be pressed before the gear can be removed from "Park". If the key is not in the ignition and the brake pedal is pressed, the Car Jacker System will be disabled and the ignition will be turned off. The microcontroller detects this

occurrence and sends a message back to the phone. This feature also ensures that a driver places the key in the ignition before departure.

## **II. Motivation**

The motivation behind the Car Jacker project was cost and educational analysis. The vehicle used to implement the project was a 2005 Toyota Camry, equipped with powerlocks but no keyless entry system. The Camry was taken to a local car alarm company where a keyless entry system was installed. The system cost approximately \$200 and did not have remote-start capabilities. Other packages had a remote-start option but cost an additional \$200 for the component. It was determined that the implementation could have been done better in terms of cost and the number of features, meanwhile providing an opportunity to understand the engineering aspect behind such a system.

Since many companies design remote starters and keyless entry, an additional motivation behind this project was to develop a system that would include aftermarket features that were not currently available. After interviewing people who have had remote-start systems installed, it was concluded that the feature most desired was the control over heating and cooling settings. The general consensus was: the settings were often left in inadequate positions for heating the vehicle - thereby defeating the purpose of the remote-starter. The prospect of adjusting the cabin temperature was deemed a useful feature to add. For example, in the freezing morning, the ability to heat the car can eliminate the need to scrape the windows. This can be beneficial to people who are not in good physical condition as well as to people who must park in uncovered areas.

## **III. Implementation**

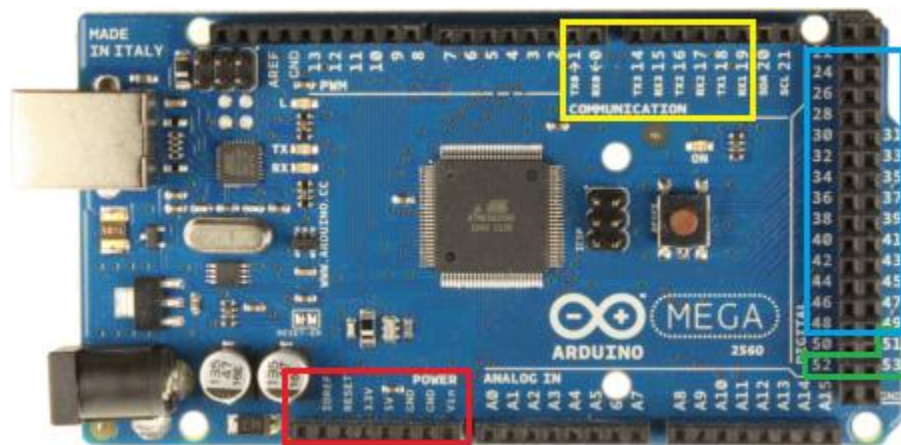
### **A-1. Hardware Components**

#### **Arduino Mega 2560**

The microcontroller used for this project is the Arduino Mega [2] which was purchased from Sparkfun.com [1]. It has 256KB of flash memory and was more than enough memory for the purpose of this project. It has four UART serial ports but only two were needed. One port was connected to the cellular shield so the phone could communicate with the microcontroller. The other port was connected to the onboard diagnostic system (OBD-II) which provided the engine status of the car. The Arduino also had plenty of digital pins (highlighted blue in Figure 1) for the relay driver circuit. The microcontroller digital signals were connected to the relay driver through optoisolators and were used for

switching the relays. The cellular shield, OBD-II, and relay driver circuit will be explained in greater detail in later sections.

The temperature control required a range of input voltages 0V - 12V to drive each temperature setting. In this case, pins 51 - 53 (MOSI, CLK, SS) of the SPI interface are used (highlighted green in Figure 1), The purpose of the SPI signals is to adjust digital potentiometers which supply variable voltages to the air mix servo motor. The Arduino is programmed by connecting to a laptop with a USB cable and installing the code. Specifications for the Arduino Mega power source recommend using 7 to 12VDC. Therefore, the 12V car battery was used to power the board. However, since the alternator can potentially boost the voltage level up to 13.8V once the car is started, a 12V voltage regulator is used to prevent the board from exceeding voltage specifications. The Arduino Mega also has a 3.3 voltage supply, administered by an onboard voltage regulator, which was used to power the cellular shield [7].

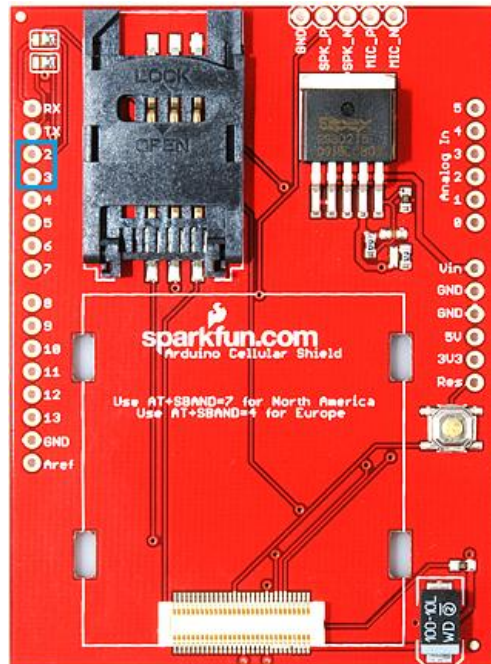


**Figure 1. Arduino Mega 2560 [2]**

### **Cellular Shield SM5100B**

The SM5100B cellular shield [7] was purchased from Sparkfun [1]. This particular cellular shield was chosen because it is able to communicate with the Android smartphone via SMS (short message service) technology. SMS is also known as “text messaging”. The shield was powered by the 3.3 voltage supply [1] from the Arduino (highlighted red in figure 1). The Arduino and cellular shield are able to transfer data by connecting one of the transmit/receive serial ports from the board (Serial-1 highlighted yellow in Figure 1) to one of the transmit/receive serial ports of the shield (highlighted blue in Figure 2). For receiving, pin 2 of the cellular shield is connected to pin 19 of the board. For transmitting, pin 3 of the cellular shield is connected to pin 18 of the board. The serial ports on the cellular shield are able to transfer up to 460 kbps. The frequency of 1900 MHz was use since it is the frequency of cellular phones in the US. In order for

the phone and cellular shield to communicate, a SIM card and an antenna are required. The SIM cards were purchased from AT&T for \$25 each and came with a 30 day unlimited text messaging plan.



**Figure 2. Cellular Shield SM5100B [7]**

### **Quad-band Wired Cellular Antenna SMA**

The quad-band wired cellular antenna sma (Figure 3) was also purchased from Sparkfun [1]. This particular antenna was chosen because it has a quad-band of 1900 MHz with a gain of 3.5 dBi [1] which can transmit and receive data from the cellular tower. It also performs well with receiving SMS messages inside a started vehicle, where noise factors have the potential to block incoming messages.



**Figure 3. Quad-band Wired Cellular Antenna SMA [1]**

## On Board Diagnostic System (OBD-II)

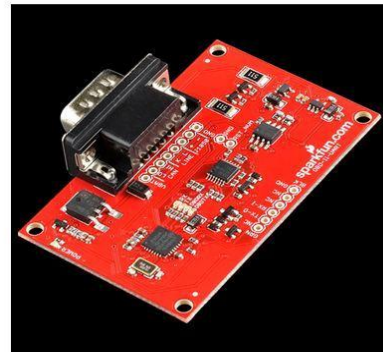
The OBD-II interface was used to report the status of the car to the Arduino board which is then used to send data to the Android. The larger end of the OBD-II cable is connected to the OBD-II port under the steering wheel and the smaller end is connected to the communication interface (part C Figure 4). The communication interface was then attached to an OBD-II UART shield which was connected to Arduino Serial-2 port. The OBD-II interface is only used to report the start/stop status of the car by reading the voltage. If the output voltage is above 13 volts, the car is started. Any lower output voltage indicates that the car is not started



A. OBD-II cable

Pin Description	OBDII	DB9
J1850 BUS+	2	7
Chassis Ground	4	2
Signal Ground	5	1
CAN High J-2284	6	3
ISO 9141-2 K Line	7	4
J1850 BUS-	10	6
CAN Low J-2284	14	5
ISO 9141-2 L Line	15	8
Battery Power	16	9

B. Pin reference

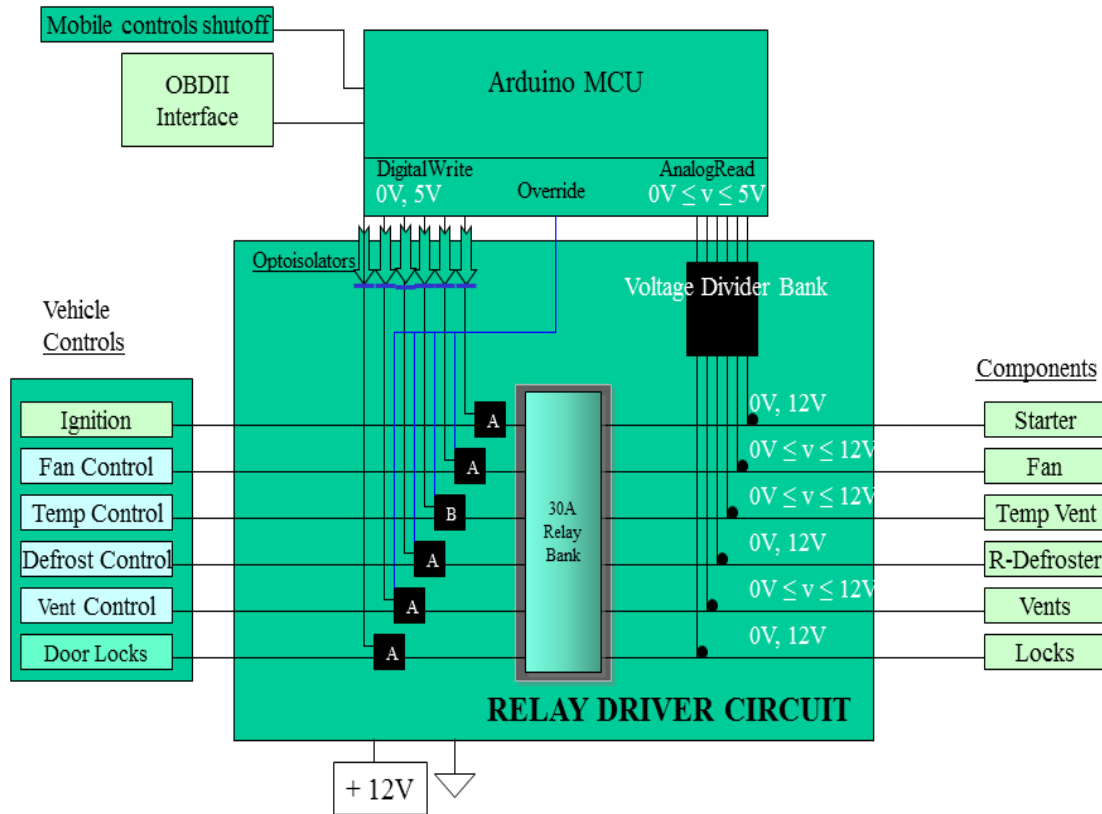


C. OBD-II Communication Interface

**Figure 4. OBD-II Interface [1]**

## Relay Driver Circuit

Each option controllable by the Car Jack system has a 3.3V-5V microcontroller digital signal parsed through the relay driver. The relay driver circuit is used to saturate the mosfet transistors which provide the 12V that is necessary to switch the relays. The relays then select between the vehicle's default controls and the Arduino driven control (these options can be observed by referencing the "vehicle controls" in Figure 5). In essence, the relays are used to bypass the power that is supplied to the dashboard switches and dials and apply the power to the settings directly. The relay driver circuit is connected to the Arduino through opto-isolators to prevent unwanted inductive feedback into the microcontroller.



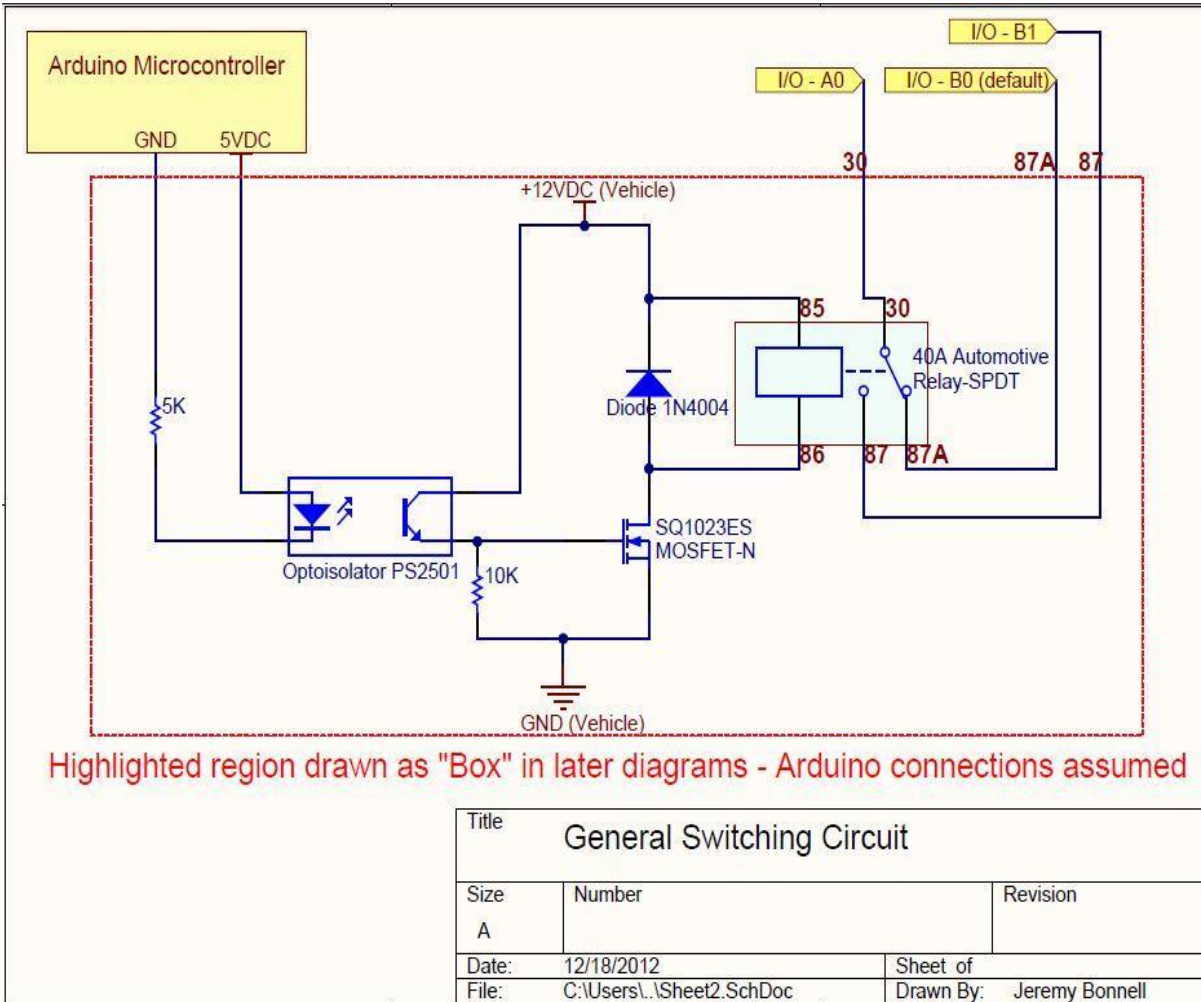
**Figure 5. Relay Overview**

There were three primary tasks the relay driver was responsible for and can be denoted as boxes 'A', 'B', and "Voltage Divider Bank" in Figure 5.

### Switch Circuit (Box 'A')

The switch circuit 'A' was used for switching override relays for the fan, vent, ignition, defrost, AC, and door lock systems. It was necessary for the car to be stopped, started, locked, or unlocked at any point in time so the input on the ignition and lock relays were always powered - except for in shutdown mode. However, the input for the heating/cooling systems is only available after the car had been started and the accessory power was available. The locks are only powered for a short period of time (1500ms), so it was not necessary to have an override switch to bypass the default locking mechanism.

The circuit of (Figure 6) is a simple MOSFET amplifier circuit and is driven by (0V,5V) digital pins from the Arduino. The transistor used is the SQ2310ES, N-Channel, 0V - 20V, 6A MOSFET by Vishay [3] (Figure 6), with a threshold voltage of 1.5V .



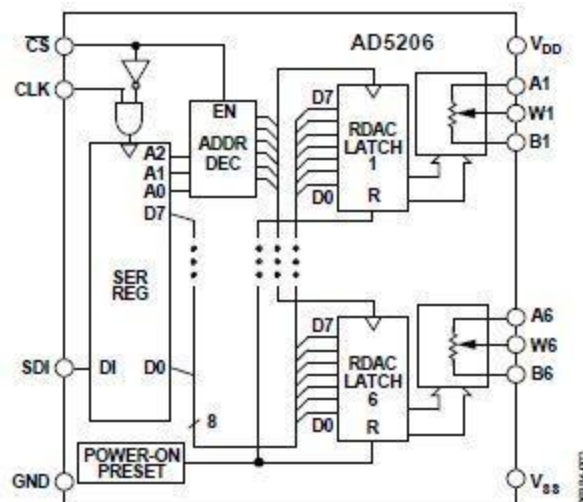
**Figure 6. Relay Driver - Switching Circuit**

The boxed area of Figure 6 signifies logic within the relay driver circuit. When the digital pin is 0V, the MOSFET behaves as an open circuit. This allows the car to be started in the normal fashion because the power is being passed through the input line (Pin 30) to the default output line (Pin 87A) on the relay. If the digital pin is set high to 5V and the ignition is not being turned, the transistor becomes saturated allowing current to flow through the transistor, and the relay is triggered. This allows the Arduino to send the input of (Pin 30) to the output line of (Pin 87). The signal is then passed into the relay bank where the setting lines can be controlled. The relay bank switches are identical to the override switches except there is nothing connected to the default (Pin 87A). Therefore, each setting can be individually turned on and off. The relay bank will be covered in more detail in a later section.

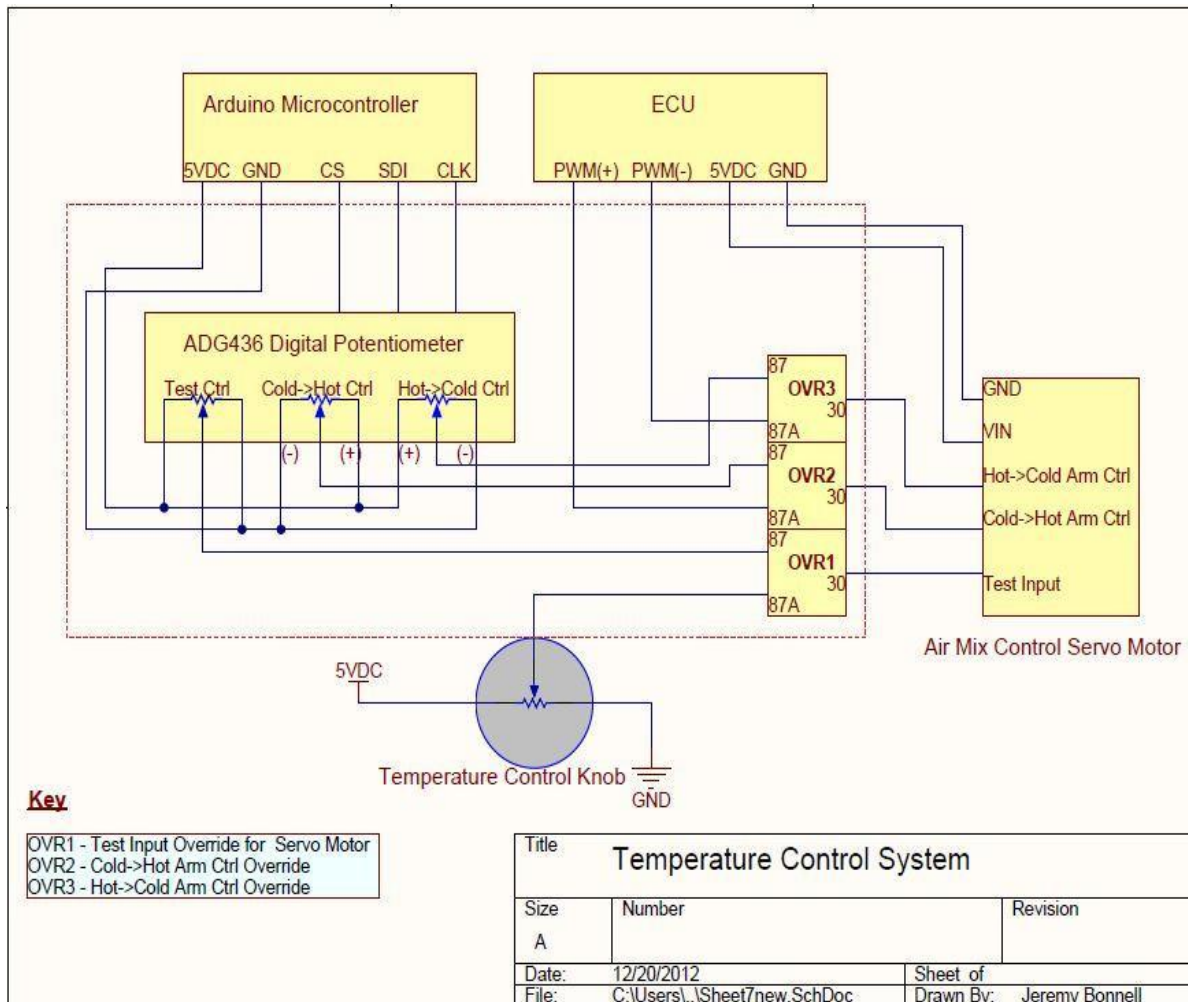


### Temperature Control Circuit (Box 'B')

The task of the temperature control circuit is to apply a variable voltage (between 0V and 12V) to temperature control air mix servo motor. With the aid of the Arduino's SPI interface, digital potentiometers were adjusted to supply the voltage to the two servo motor control signals. If one of the control signals was set to 12V and the other to ground, the arm would position the vent to be moved to the heater core. If the voltage and ground were switched, the servomotor arm would be positioned to the cooling core. Two digital potentiometers were moved in unison to achieve this task. Opposing sides of the potentiometers received the 12V, while the other opposing sides received the ground signal. The wiper pin was used to send the variable voltages to the servo motor control signals. In addition, an air mix damper position sensor pin had to be attached to one of the wiper pins to tell the motor when to stop. The digital potentiometers used (Figure 7) is the AD5206 from Analog Devices [4], with 10K $\Omega$  terminal resistance and 256 positions and 6 On-Chip potentiometers.



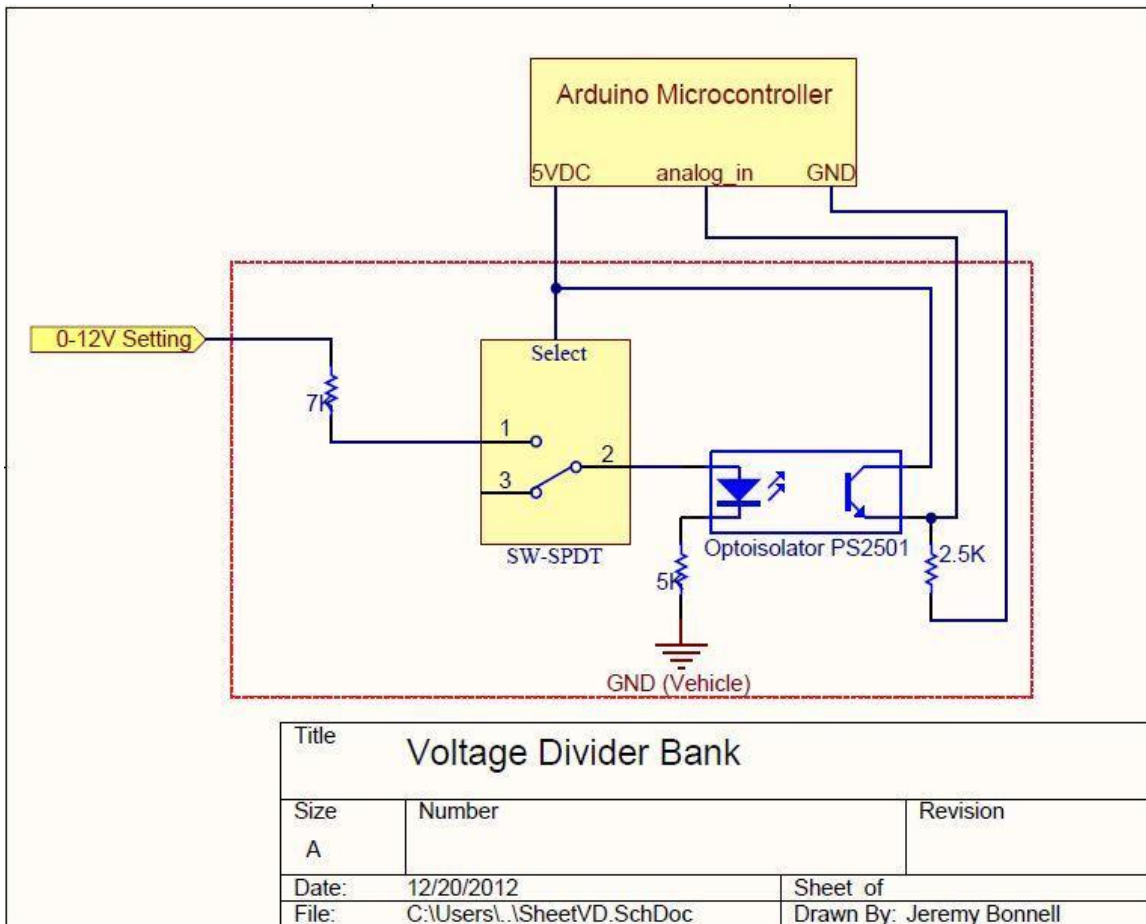
**Figure 7. AD5206 Digital Potentiometer [4]**



**Figure 8. Relay Driver - Temperature Control Circuit**

### Voltage Divider Bank

This portion of the circuit is a set of voltage dividers that lower switch output voltages to the Arduino's 5V input specifications. Each voltage divider, (Figure 9) consists of a 7K $\Omega$  and 5K $\Omega$  resistor in series with the 5K $\Omega$  resistor leading to ground. Between the two resistors is a connection leading to an analog\_in port on the Arduino. This causes the maximum switch output of 12V to be reduced to 5V and safely be used as Arduino input. The Arduino typically does not poll these ports unless a call is made to read the voltage from a particular pin in software. Once an analogRead() call is made in software, the analog port allows voltage to pass long enough to take a "snapshot" of the line voltage, the line is set back to an open circuit, and an integer of 0-1023 is returned in software.



**Figure 9. Relay Driver - Single Voltage Divider**

The integer can then be translated and returned to the Android to display the vehicle's current settings. Also within the voltage divider circuit is another ADG1436 switch positioned between the 5KΩ resistor and ground. One input of the switch is connected to nothing, while the other input is connected to the 5KΩ resistor. The output is connected to ground. Using a digital pin from the Arduino, the "Div\_select" line is set to default on the nothing input. Right before the call to analogRead() the select line is set to the 5KΩ input then set back to nothing afterwards. This keeps the branch from being a continuous drain on the switch output.

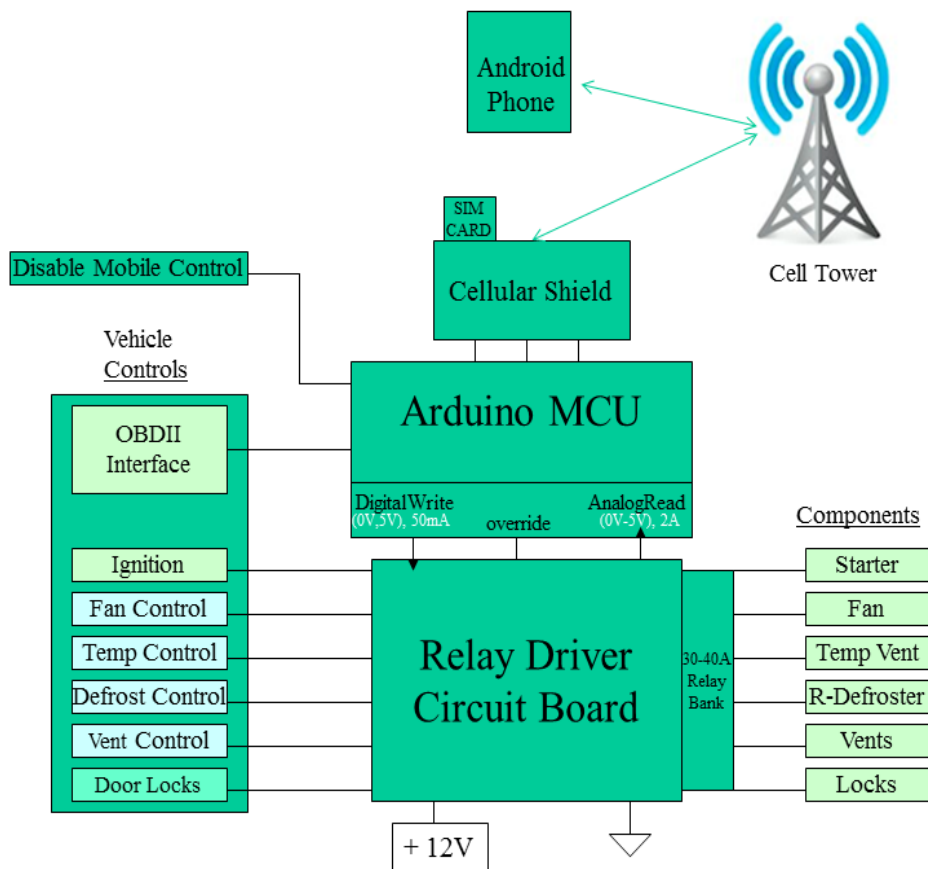
### Relay Bank

The relay bank is an array of the switch circuits seen in (Figure 6) and is used to turn individual settings on and off. The relays can essentially be thought of as multiplexers with nothing connected to the default output lines. When the override signal is low, the relay bank is not receiving any setting signals. When the override signal is high, the relay bank receives the power that was bypassed from all the default controls. Each individual system, such as ignition system, fan system, etc., has its own network within the relay bank. When power is being supplied to the relay bank, individual settings for

each system can be turned on and off. The vehicle's starter motor, blower motor, and locks required up to 40A of current to operate. The relays used by the Car Jacker are the 275-266 SPDT 30A and 40A Automotive Grade Relays from Radio Shack [6]. Diagrams for the networks within the relay bank can be viewed near the end of the document and precedes the source code.

## A-2. Hardware Procedure

Using the SMS (Short Message Service) capabilities installed on the Android smartphone, a signal is transmitted to the cellular shield and the data is parsed to the Arduino microcontroller. OBD-II data is also sent to the microcontroller to determine if the engine is on or off. The Car Jack system is installed in the Camry by splicing into the vehicle's heating/cooling, door lock, and ignition systems (Wiring for individual settings can be found at the end of the document). This allows the microcontroller to capture and translate the vehicle's current settings and transfer them to the phone where a GUI displays them to the user. A general overview of the design can be seen in Figure 10.



**Figure 10. Design Overview**

Some components within the vehicle's wiring scheme, such as vent and temp settings, are controlled by servo motors. Therefore in order to parse the signals through the microcontroller, the splicing of the heating/cooling, lock, and ignition wires are done at the back of the dashboard controls. This required a fair amount of reverse engineering since the setting of every controller's voltage had to be recorded in order for the microcontroller to decipher the settings they are currently on. Each setting has a power signal that is sent to either a switch or a dial. When the dial is turned or a switch is pressed, the circuit is completed and the starter motor, blower motor, etc... is powered. The different settings for the motors are basically a range of voltages. For a switch, such as the rear defrost, this is trivial. The switch simply makes a direct connection from the power line to the rear defrost heating element. For servo motors or blower motor, the settings are more complex. There are either an array of discrete signals that control the different voltages required for each setting, or there are potentiometers with a single line used to supply a whole range of voltages.

To determine what the current setting is for the discrete configurations or a switch configuration, a direct reading can be sent into the microcontroller. Each setting was spliced from the individual setting lines and sent to a digital input line of the microcontroller. If that line is high, this is the setting that is on.

To determine what the current setting is for the potentiometer configuration, an analog read must be sent into the microcontroller's "analog\_in" pins. This is an Analog to Digital Conversion (ADC). This is where the microcontroller takes a voltage of an analog signal and converts it into an integer value between (0-1023) for (0V-5V) within the software of the Arduino. The voltage divider bank is used to lower 12V signals to the specification of 5V. Every setting was measured with a multimeter to determine what the voltage was for that particular setting. For the temperature, there were 31 settings. Within the software, each value for all the settings is defined. To determine the current setting, the Arduino polls the analog\_in pin, converts it to an integer value, and cross-references the settings. The current setting can then be sent to the Android for the user to view.

Information from reverse engineering was also used to determine which settings were needed for the relays. Within the microcontroller, the software implementation is responsible for allowing direct or Arduino control of the settings. For Arduino control, this was done by setting the override signal to high and adjusting the digital potentiometer values to control voltage levels within the relays and switch in between the Arduino override control and default controls. In addition, each signal from the relay has an optoisolator attached to prevent reverse EMF (electromagnetic flux) which could potentially destroy the microcontroller circuit.

The phone initializes the system by requesting the vehicle start/stop status. The OBD-II interface first checks the ignition status of the vehicle and sends it to the Arduino to be returned to the phone. If the car has not been started, the option to start the car will be enabled. Otherwise the heating/cooling, door lock, and start status of the car will be fetched and returned to phone. At this point the option to override the heating/cooling systems will be given to the user. If an override signal is sent by the phone, the Arduino enables the override signal to the relays. The dashboard heating/cooling settings are then switched to the Arduino driven controls with the SPDT switch. The Arduino then sends digital signals to power the relay circuits and the SPI output are used to set the digital potentiometers. The Arduino settings are initially set the same as the dashboard settings, but can now be adjusted via the Android app. If the vehicle is not being driven, the door lock and ignition functions are enabled. Heating/cooling functions are available if the car is started and the vehicle is in park. These detections are done by the moving the shift knob out of park. Wiring for individual settings can be found at the end of the document.

## **B-1. Software Implementation**

The software implementation of this project was a GUI interface written on an Android smartphone and displayed control options and vehicle's start and heating/cooling status to the phone. It generated encoded strings to be sent to the microcontroller via text messaging. This required the use of a SIM card in order to transmit and receive the SMS signals. Once the signal has been received and decoded by the microcontroller, a confirmation signal sends back to the phone and the phone displays the current status to the screen. Both the phone and microcontroller needed to know the encoding scheme used in order to communicate effectively.

### **Arduino**

The Arduino microcontroller parses messages sent from the phone and performs the appropriate task based on message details. The Arduino task list is labeled in (Figure 11). The majority of tasks are done by setting digital pins to HIGH or LOW, which in turn, switch relays. The temperature control system requires a range of voltages for its numerous settings. Digital potentiometers are used to adjust these levels. The digital potentiometers are operated with the Arduino SPI interface, pins 51-53. A tutorial on digital potentiometers is available on Arduino's website [2]. The Arduino also receives signals from the OBD-II to retrieve the engine status of the vehicle. Messages are sent to the phone by means of the cellular shield mounted on the Arduino. The Arduino communicates with the cellular shield by sending AT commands along the serial ports. When the Arduino sends a message to the phone, it sends the appropriate AT command

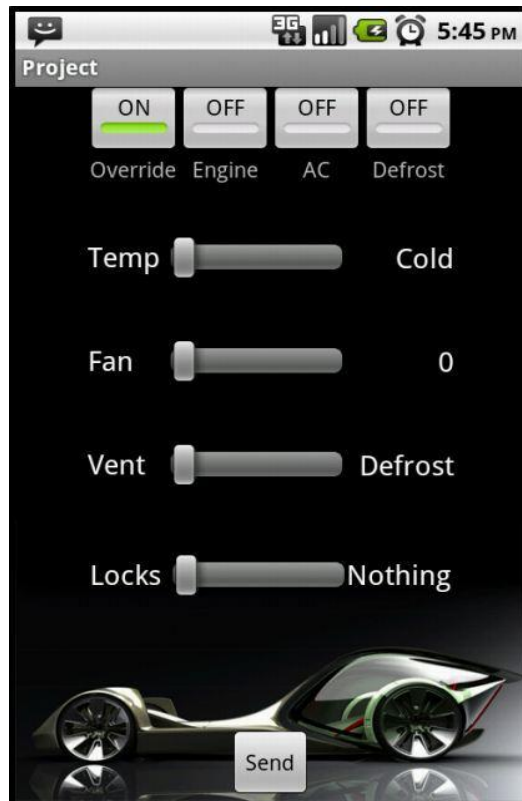
the cellular shield, along with the Android mobile number and the message itself. All the programming for the board was written using Arduino 1.0 IDE.

### Android Application

The phone messages are sent to the mobile number of an activated SIM card. The SIM card is mounted on the cellular shield and its mobile number is hard coded in software. The messages are sent internally with SMS (short message service) and were linked to buttons in the Android GUI application. When the “Send” button is pressed, it sends the message to the Arduino. Each message consists of 7 characters with the first 2 always being “zz”. These characters indicate the initialization sequence to the system and filter out unwanted text messages. The 4 remaining characters are represented as bits of hexadecimal characters. Each hex bit is comprised of 4 binary bits and each binary bit represents a setting. A chart of how a message is constructed is shown in figure 11. For example, “zza7812” corresponds to lock door, turn on AC, turn on override, turn on ignition/engine, turn on rear defrost, set fan to level 4, set vent to level 1 (defrost), and set temperature to hot. The Arduino microcontroller also sends messages to the phone for the status of the car. All the code of the Android application was written in Eclipse IDE with Android SDK (Android Software Development Kit). An image of what the GUI looks like is shown in (Figure 12).

	Char 1	Char 2	Char 3	Char 4	Char 5 (Fan levels)	Char 6 (Vent levels)	Char 7 (Temp levels)
Bit 0	z	z	Not Used	Override	Fan level 1	Vent level 1	Hot
Bit 1	z	z	AC	Engine	Fan level 2	Vent level 2	Cold
Bit 2	z	z	Unlock door	Defrost	Fan level 3	Vent level 3	Not Used
Bit 3	z	z	Lock door	Not Used	Fan level 4	Vent level 4	Not Used
Example	Ex: z	Ex: z	Ex: a	Ex: 7	Ex: 1	Ex: 4	Ex: 2

**Figure 11.** Bit representation for settings.



**Figure 12. Android GUI**

## IV. Tasks

The first task accomplished was the gathering of materials and components for the project. A breakdown of all items necessary for the project is as follows:

- Arduino Microcontroller Board
- Cellular Shield
- \* Vehicle
- Android Smartphone
- Multimeter
- SIM card
- Antenna
- OBD-II Plug
- OBD-II Shield
- 2005 Toyota Camry wiring diagrams and schematics
- (22, 16, 14, 12, 10) AWG wire. (~250 ft. total)
- 10K $\Omega$ , 5K $\Omega$ , 2K $\Omega$  (1/4 Watt) Resistors
- Diodes
- Optoisolators



- Wire nuts, slice connectors, Anderson Powerpole connectors
- N-Channel Mosfet Transistors
- Digital Potentiometers
- 30-40A - Automotive grade Relays (SPDT Switch)
- 5A SPDT - (DIP) (for breadboard small circuits)

Gathering components was crucial since the initialization of the project could not begin until the communication components were acquired. The Arduino, Cellular Shield, antenna, and the first SIM card were all ordered by mid-April. Communication between the cellular shield and Arduino were complete by the end of Spring Semester 2012

The next task worked on was the sending and receiving of the SMS signal. This task began by implementing a simple Android application that allowed the phone to send a short SMS signal to the transmitter/receiver attached to the microcontroller. The signal performed trivial tasks, such as turning on an LED. Then the advancement of more complicated tasks were initialized, such as - sending a sequence of signals, light a sequence of LED's, return a signal to the phone and display something to the screen. After the signals were being sent and interpreted correctly, an Android application was written to interact with the hardware. This step required minimal wiring and was not the most time consuming portion of the project. The most time consuming portion of the project was the building the relay driver circuit board and the wiring of relays to the vehicle. By the end of Summer semester - August 2012, the relay driver circuit board was constructed, the relays were wired to the board and could be switched on and off, and the first iteration of software for the Android and Arduino were written and ready for initial testing.

The final task was the reverse engineering and hardware connections between the systems of the vehicle. The dashboard of the vehicle had to be removed to access the wiring of the controls. Next, each setting had to have their voltage read with a multimeter and recorded for the software implementation. Each setting was then wired to through the relays and tested individually until all settings were complete. The order of systems completed was the locks, fan, vent, rear defrost, ignition, temp, and shutdown system. The goal was to complete the project one month before the "demo day" exhibit in Fall semester 2012. We have successfully accomplished all goals by demo day.

## **V. Risks and Interface Issues Encountered**

- The transistor and relay switch circuits took much longer to design than anticipated.

- Many small test circuits had to be built on breadboards and tested in the vehicle, under actual circumstances, to make sure they would not fry. Several components were not grounded and did in fact fry. A breadboard was also melted.
- The vehicle brake and OBD-II fuses were blown several times.
- If an override was placed on the ignition switch, it was difficult to check if the key was inserted. A separate custom circuit had to be built through the ignition switch, while the default power was not supplied, to enable the vehicle shutdown mode.
- The servo motor for the temperature setting was difficult to figure out. The 2005 Camry diagrams found on [www.alldata.com](http://www.alldata.com) [8] did not show PWM signals that were connected to the vehicle's ECU. This system took the longest to reverse engineer.
- The encoded messages sent from the Android had to be reconfigured to hexadecimal. The cellular shield would not always consistently pass the entire text message. Message passing was constrained to 7 characters to make sure the system was reliable.

## VI. Lessons Learned

The risks were subject to, like everyone else, running the risk of not receiving all materials in time. Also, the OBD-II and relays, and signal interpretation proved to be difficult. We also had to be careful about dismantling the vehicle and its wiring. If caution is not practiced the dashboard or one of us could have gotten hurt. We also needed to make diagrams so we did not detach a bunch of wires and not know how to rewire them. With only a two-man group, we also ran the risk of not having enough man-power to complete the project. This is why we needed to start as early as possible. We had to work very hard to complete this project in time! But in the end, the challenge and learning experience were well worth it.

## VII. Tasking and Scheduling

1. Communication between Phone and Arduino	3 weeks
2. Android Application and Arduino code	2 weeks
3. Relay Circuit	2 week
3. Connections: Car with Relay	4 weeks
4. Connections: Arduino and Relay	1 week

- |  |         |
|--|---------|
| 5. Communication between Phone and Car           | 4 weeks |
| 6. Finishing touches: Controls from Phone to Car | 4 weeks |

Testing/Debugging every step at a time

## VIII. Bill of Materials

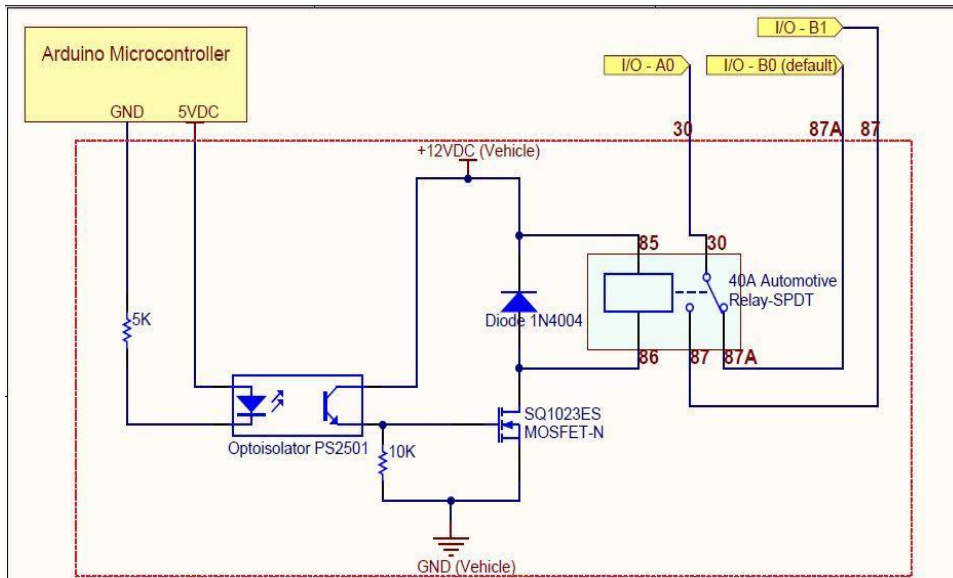
Toyota Camry	\$...
Android Phone	\$...
Sim Card (3) from AT&T	\$75
Arduino Mega 2560 (1) from Sparkfun	\$70
Cellular Shield SM5100B (1) from Sparkfun	\$100
Relays (20) from automate (Resistors (20), Potentiometers (2), Mosfets (20), Diode (20), Optoisolator (20), Splicers (40))	\$150
OBD-II RS232 interface and OBD-II UART shield (1) from sparkfun	\$60
Antenna (1) from Sparkfun	\$20

## IX. Bibliography

### References

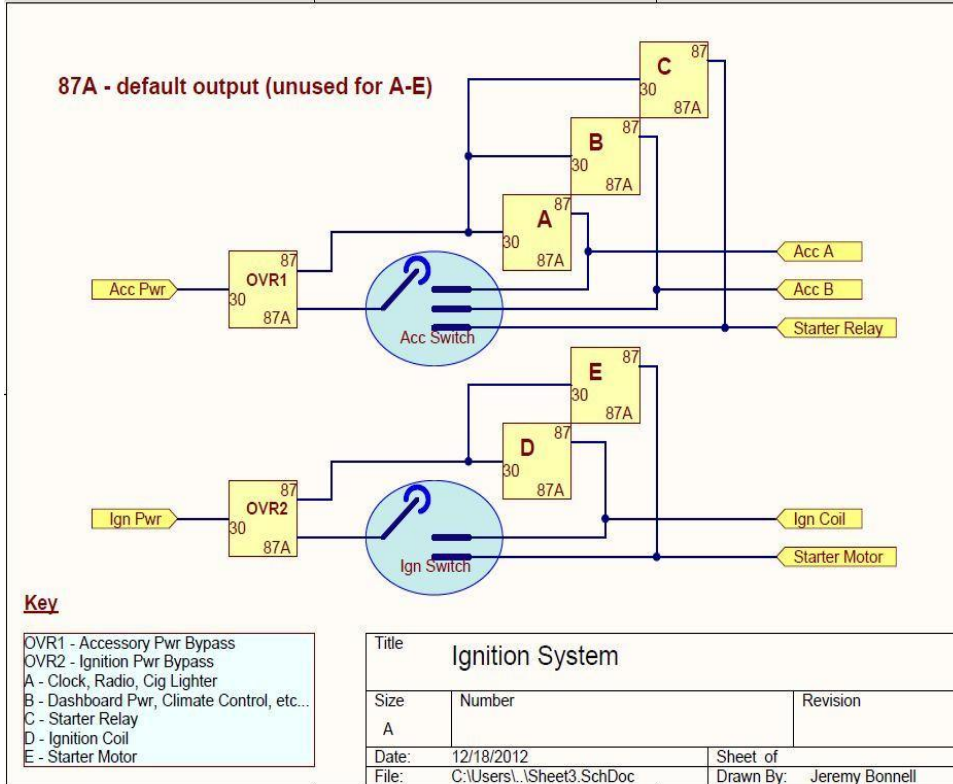
1. Sparkfun. [Online] April 21, 2012. [Cited: April 21, 2012.] <http://www.sparkfun.com/>
2. Arduino. [Online] April 21, 2012. [Cited: April 21, 2012.] <http://arduino.cc/en/>
3. Vishay. [Online] April 21, 2012. [Cited: April 21, 2012.] . N-Channel MOSFET  
<http://www.vishay.com/docs/67036/sq2310es.pdf>
4. Analog Devices. [Online] May 1, 2012. [Cited: May 1, 2012.] .AD5206 Digital Pot.  
[http://www.analog.com/static/imported-files/data\\_sheets/AD5204\\_5206.pdf](http://www.analog.com/static/imported-files/data_sheets/AD5204_5206.pdf)
5. Analog Devices. [Online] April 21, 2012. [Cited: April 21, 2012.] . SPDT Switch  
[http://www.analog.com/static/imported-files/data\\_sheets/ADG1436.pdf](http://www.analog.com/static/imported-files/data_sheets/ADG1436.pdf)
6. RadioShack. [Online] May 1, 2012. [Cited: May 1, 2012.] .SPDT 30A Automotive  
Grade Relay. <http://www.radioshack.com/product/index.jsp?productId=2062477>
7. Cellular Shield - Sparkfun. [Online] April 21, 2012. [Cited: April 21, 2012.]  
<http://www.sparkfun.com/products/9607>
8. AllData. [Online] May 1, 2012. [Cited: May 1, 2012.] .2005 Toyota Camry Diagrams.  
<http://www.alldata.com>

## X. System Diagrams



Highlighted region drawn as "Box" in later diagrams - Arduino connections assumed

Title			General Switching Circuit		
Size	Number		Revision		
A					
Date:	12/18/2012		Sheet of		
File:	C:\Users\...\Sheet2.SchDoc		Drawn By: Jeremy Bonnell		

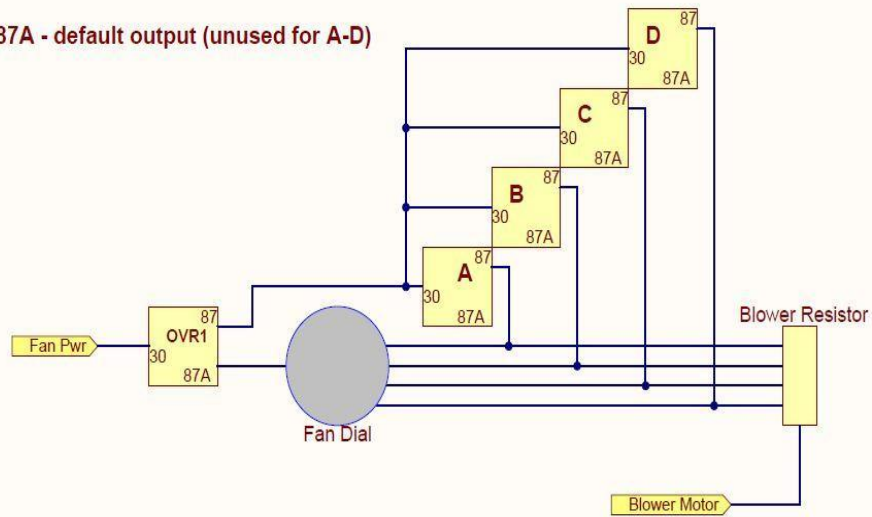


**Key**

- OVR1 - Accessory Pwr Bypass
- OVR2 - Ignition Pwr Bypass
- A - Clock, Radio, Cig Lighter
- B - Dashboard Pwr, Climate Control, etc...
- C - Starter Relay
- D - Ignition Coil
- E - Starter Motor

Title			Ignition System		
Size	Number		Revision		
A					
Date:	12/18/2012		Sheet of		
File:	C:\Users\...\Sheet3.SchDoc		Drawn By: Jeremy Bonnell		

87A - default output (unused for A-D)

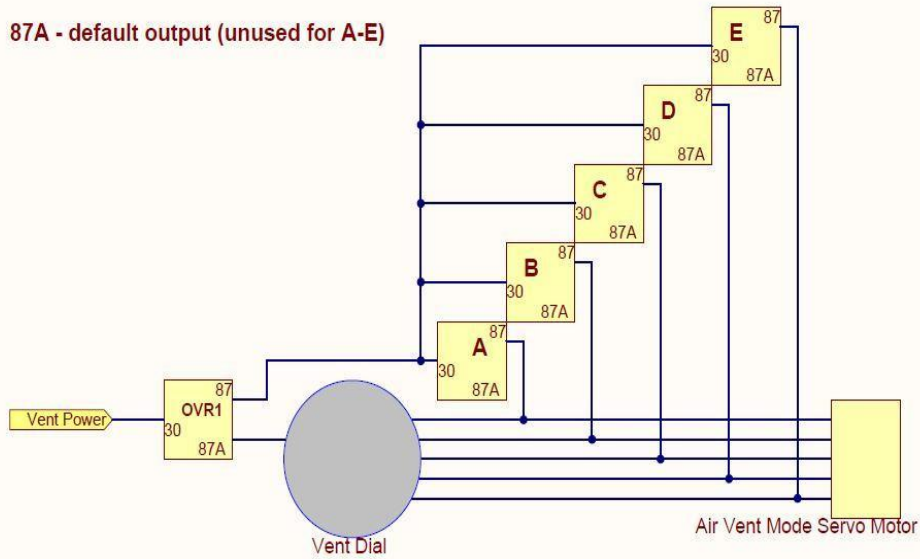


**Key**

- OVR1 - Fan Pwr Bypass
- A - Low Setting
- B - Med/Low Setting
- C - Med/High Setting
- D - High Setting
- E - Face

Title Fan System		
Size A	Number	Revision
Date: 12/18/2012	Sheet of	
File: C:\Users\...\Sheet4.SchDoc	Drawn By: Jeremy Bonnell	

87A - default output (unused for A-E)

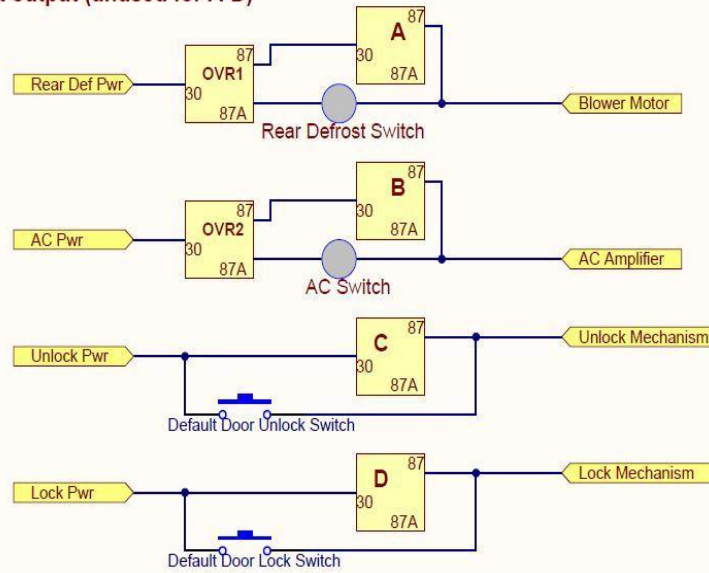


**Key**

- OVR1 - Vent Pwr Bypass
- A - Defrost
- B - Foot/Def
- C - Foot
- D - Face/Foot
- E - Face

Title Vent System		
Size A	Number	Revision
Date: 12/18/2012	Sheet of	
File: C:\Users\...\Sheet5.SchDoc	Drawn By: Jeremy Bonnell	

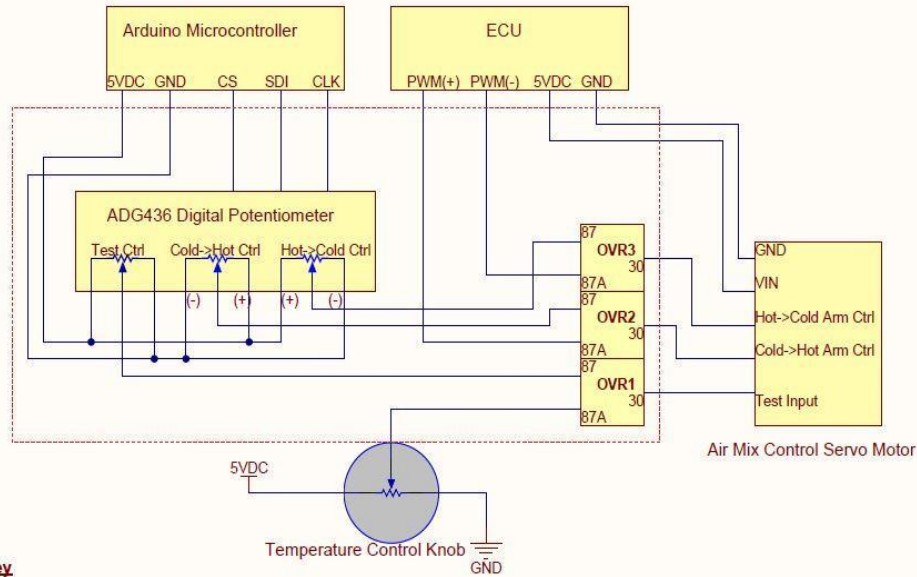
**87A - default output (unused for A-D)**



**Key**

OVR1 - RDef Pwr Bypass  
 OVR2 - AC Pwr Bypass  
 A - RDef Relay Switch  
 B - AC Relay Switch  
 C - Door Unlock Relay Switch  
 D - Door Lock Switch

Title Rear Defrost, AC, and Door Lock Systems		
Size A	Number	Revision
Date: 12/18/2012	Sheet of	
File: C:\Users\...\Sheet6.SchDoc	Drawn By:	Jeremy Bonnell



**Key**

OVR1 - Test Input Override for Servo Motor  
 OVR2 - Cold->Hot Arm Ctrl Override  
 OVR3 - Hot->Cold Arm Ctrl Override

Title Temperature Control System		
Size A	Number	Revision
Date: 12/20/2012	Sheet of	
File: C:\Users\...\Sheet7new.SchDoc	Drawn By:	Jeremy Bonnell

## XI. Code

### Arduino Code

```
#include <string.h>      //Used for string manipulations
#include <SPI.h>

char in_char=0;    //Will hold the incoming character from the Serial Port.
char mobilenumber[] = "8019108604"; //phone number
boolean checker = false;
boolean override = false;
boolean ready = false;
boolean engineOn = false; //See if the engine is on or off
boolean notified = true; //Notification when phone received message
int currentState = 0;
int counter = 0;
boolean checkBreak = false;

const int inPin = 7; //Reading pin for break, key, shift. If this is high, everything off.
const int outPin = 6; //Constant high pin
int val = 0; //temp val for shift knob

//Fan pinouts
const int fanOvr = 22;
const int fanA = 24;
const int fanB = 26;
const int fanC = 28;
const int fanD = 30;
int fanPins[] = {fanOvr, fanA, fanB, fanC, fanD};

//Engine pinouts
const int engOvr1 = 31;//acc
const int engOvr2 = 32;//ign
const int engA = 33;
const int engB = 34;
const int engC = 35;
const int engD = 36;
const int engE = 37;
int enginePins[] = {engOvr1, engOvr2, engA, engB, engC, engD, engE};
```

```

//Vent pinouts
const int ventOvr = 38; //Override for all
const int ventA = 39;//def
const int ventB = 40;//foot/def
const int ventC = 41;//foot
const int ventD = 42;//face/foot
const int ventE = 43;//face
int ventPins[] = {ventOvr, ventA, ventB, ventC, ventD, ventE};

//Lock pinouts
const int lockOvr = 44;//unused
const int lockA = 45;
const int lockB = 46;
int lockPins[] = {lockOvr, lockA, lockB};

//Def pinouts
const int defA = 48;
int defPins[] = {defA};

//Temp pinouts
const int tempOvr = 49; //Override
const int tempAC = 47; //AC
int tempPins[] = {tempOvr, tempAC};

const int tempCold = 0; //Temp cold (INT ADDRESS OF DIGIPOT-1 ON CHIP)
const int tempHot = 2; //Temp hot (INT ADDRESS OF DIGIPOT-2 ON CHIP)
const int slaveSelectPin = 53; //SPI pin

//-----Initialization-----
void setup()
{
  //Reset all pins
  ResetAll();

  //Initializing read and constant pins for break, key, shift signal
  pinMode(outPin, OUTPUT);
  pinMode(inPin, INPUT);
  digitalWrite(outPin, HIGH);
  // set the slaveSelectPin as an output:
  pinMode (slaveSelectPin, OUTPUT);
}

```



```
// initialize SPI:
SPI.begin();

delay(100);
for (int channel = 0; channel < 6; channel++) {
    setTemp(channel, 0);
}

//Initialize serial ports for communication.
Serial.begin(9600);
Serial1.begin(9600);
Serial2.begin(9600);
Serial.flush();
Serial2.flush();
delay(10000);
Serial.println("Starting SM5100B Communication...");
while(ready == false)
{
    if(Serial1.available())
    {
        delay(10);
        in_char = Serial1.read();
        Serial.print(in_char);
        if(in_char == 'S')
        {
            delay(10);
            in_char = Serial1.read();
            Serial.print(in_char);
            if(in_char == 'l')
            {
                delay(10);
                in_char = Serial1.read();
                Serial.print(in_char);
                if(in_char == 'N')
                {
                    delay(10);
                    in_char = Serial1.read();
                    Serial.print(in_char);
                    if(in_char == 'D')
                    {
```

```

    delay(10);
    in_char = Serial1.read();
    Serial.print(in_char);
    if(in_char == ':')
    {
        delay(10);
        in_char = Serial1.read();
        Serial.print(in_char);
        if(in_char == ' ')
        {
            delay(10);
            in_char = Serial1.read();
            Serial.print(in_char);
            if(in_char == '4')
            {
                delay(10);
                in_char = Serial1.read();
                Serial.print(in_char);
                Serial.println();
                Serial.println("Ready for action");
                ready = true;
            }
        }
    }
}

//Serial1.println("AT+SBAND=7");
//Serial1.println("AT+CFUN=1,1"); //reset shield
//delay(1000);
//Serial1.println("AT+IPR?"); //baud rate, has been set
Serial1.println("AT+CMGF=1"); // set SMS mode to text
Serial1.println("AT+CNMI=3,3,0,0"); // set module to send SMS data to serial out upon
receipt
Serial1.println("AT+CMGD=1,4"); // delete all SMS
//Wait for a little while before sending the reset command to the OBD-II-UART
delay(1500);

```

```
//Reset the OBD-II-UART
Serial2.print("ATZ\r");
//Wait for a bit before starting to send commands after the reset.
delay(1000);
Serial2.flush();
delay(2000);
Serial.println("Ready");
}
```

```
//-----Running-----
void loop()
{
  //Checking if the engine is on
  if(checker == false)
  {
    checkBreak = false;
    val = digitalRead(inPin);
    if(val == 0 && currentState == 0)
    {
      Serial.println("Break: System Off");
      Override(LOW);
      OverrideEngine(LOW);
      ResetAll();
      override = false;
      notified = false;
      engineOn = false;
      currentState = 1;
      checkBreak = true;
      checker = true;
      delay(5000);
    }
    else if(val == 1 && currentState == 1)
    {
      Serial.println("Unbreak: System Ready");
      currentState = 0;
      checker = true;
      checkBreak = true;
      delay(5000);
    }
  }
}
```

```
if(checkBreak == false)
{
//Receive 1 text message
while(Serial1.available() > 0)
{
  delay(10);
  in_char = Serial1.read();
  Serial.print(in_char);

  if(in_char == 'S')
  {
    delay(10);
    in_char = Serial1.read();
    Serial.print(in_char);

    if(in_char == 'I')
    {
      delay(10);
      in_char = Serial1.read();
      Serial.print(in_char);
      if(in_char == 'N')
      {
        delay(10);
        in_char = Serial1.read();
        Serial.print(in_char);
        if(in_char == 'D')
        {
          delay(10);
          in_char = Serial1.read();
          Serial.print(in_char);

          if(in_char == ':')
          {
            delay(10);
            in_char = Serial1.read();
            Serial.print(in_char);

            if(in_char == ' ')
            {
```

```

    delay(10);
    in_char = Serial1.read();
    Serial.print(in_char);

    if(in_char == '4')
    {
        delay(10);
        in_char = Serial1.read();
        Serial.print(in_char);

        Serial1.println("AT+CMGF=1"); // set SMS mode to text
        Serial1.println("AT+CNMI=3,3,0,0"); // set module to send SMS data to
serial out upon receipt
        Serial1.println("AT+CMGD=1,4"); // delete all SMS
        delay(1000);
    }
    }
    }
    }
    }
}

else if(in_char == 'T')
{
    delay(10);
    in_char = Serial1.read();
    Serial.print(in_char);
    if(in_char == 'I')
    {
        delay(10);
        in_char = Serial1.read();
        Serial.print(in_char);
        if(in_char == 'N')
        {
            ready = false;
            Serial1.println("AT+CFUN=1,1"); //reset shield

            while(ready == false)
            {
                if(Serial1.available())

```

```
{
  delay(10);
  in_char = Serial1.read();
  Serial.print(in_char);
  if(in_char == 'S')
  {
    delay(10);
    in_char = Serial1.read();
    Serial.print(in_char);
    if(in_char == 'I')
    {
      delay(10);
      in_char = Serial1.read();
      Serial.print(in_char);
      if(in_char == 'N')
      {
        delay(10);
        in_char = Serial1.read();
        Serial.print(in_char);
        if(in_char == 'D')
        {
          delay(10);
          in_char = Serial1.read();
          Serial.print(in_char);
          if(in_char == ':')
          {
            delay(10);
            in_char = Serial1.read();
            Serial.print(in_char);
            if(in_char == ' ')
            {
              delay(10);
              in_char = Serial1.read();
              Serial.print(in_char);
              if(in_char == '4')
              {
                delay(10);
                in_char = Serial1.read();
                Serial.print(in_char);
                Serial.println();
              }
            }
          }
        }
      }
    }
  }
}
```

```

        Serial.println("Ready for action");
        ready = true;
    }
}
}
}
}
}
}
}
}
}
Serial1.println("AT+CMGF=1"); // set SMS mode to text
delay(200);
Serial1.println("AT+CNMI=3,3,0,0"); // set module to send SMS data to serial
out upon receipt
delay(200);
Serial1.println("AT+CMGD=1,4"); // delete all SMS
}
}
}
}
//End of a text message
/*if(in_char == '\r')
{
    Serial.println("End message");
    checker = true;
    break;
}*/

//Every text message has 7 characters begin with zz and follow by xxxxx hex values
//Begin
if(in_char == 'z')
{
    delay(10);
    in_char = Serial1.read();
    Serial.print(in_char);
    if(in_char == 'z')
    {
        checker = true;
    }
}

```

```
Serial2.print("ATRV\r");
while(Serial2.available())
{
  delay(10);
  in_char = Serial2.read();
  if(in_char == '1') {
    delay(10);
    in_char = Serial2.read();
    if(in_char == '2') {
      Serial.println("Car not started");
      //Car has not started
      //notified = false;
      engineOn = false;
      break;
    }
  }
  else if(in_char == '3') {
    Serial.println("Car started");
    engineOn = true;
    /*
    //car started
    val = digitalRead(inPin);
    //System off state
    if(val == 1)
    {
      Serial.println("System Off");
      Override(Low);
      OverrideEngine(Low);
      ResetAll();
      override = false;
      notified = false;
      engineOn = false;
      checker = true;
    }
    //Ready to go
    else if(val == 0)
    {
      Serial.println("Ready to go");
      notified = false;
      engineOn = true;
      checker = true;
    }
  }
}
```



```
    */
    break;
}
}
}
Serial2.flush();

if(val == 0)
{
    break;
}
delay(1000);
Serial.println("3rd Character");
in_char = Serial1.read();
Serial.print(in_char);

//Start parsing message
//3rd character
//bit 0 indicates notified bit
//bit 1 nothing potentially AC
//bit 2 indicates Unlockdoor
//bit 3 indicates Lockdoor
switch (in_char)
{
    case '0': //0000
        notified = false;
        ACOff();
        break;
    case '1': //0001
        notified = true;
        ACOff();
        break;
    case '9': //1001
        notified = true;
        LockDoor();
        ACOff();
        break;
    case '8': //1000
        notified = false;
        LockDoor();
```

```
    ACOff();
    break;
case '4': //0100
    notified = false;
    UnlockDoor();
    ACOff();
    break;
case '5': //0101
    notified = true;
    UnlockDoor();
    ACOff();
    break;

case '2': //0010
    notified = false;
    ACOff();
    break;
case '3': //0011
    notified = true;
    ACOff();
    break;
case 'b': //1011
    notified = true;
    LockDoor();
    ACOff();
    break;
case 'a': //1010
    notified = false;
    LockDoor();
    ACOff();
    break;
case '6': //0110
    notified = false;
    UnlockDoor();
    ACOff();
    break;
case '7': //0111
    notified = true;
    UnlockDoor();
    ACOff();
```

```
    break;
}

//4th character
//bit 0 indicates override  1 for on 0 for off
//bit 1 indicates engine
//bit 2 indicates defrost
//bit 3 nothing
delay(1000);
Serial.println("4th Character");
in_char = Serial1.read();
Serial.print(in_char);
switch (in_char)
{
  case '0':  //0000
    Override(LOW);
    ACOff();
    delay(6000);
    OverrideEngine(LOW);
    ResetAll();
    override = false;
    engineOn = false;
    break;
  case '1':  //0001
    if(override == false)
    {
      Override(HIGH);
      OverrideEngine(HIGH);
      override = true;
    }
    if(engineOn == true)
    {
      DefrostOff();
      StopEngine();
      engineOn = false;
    }
    break;
  case '3':  //0011
    if(override == false)
    {
```

```
Override(HIGH);
OverrideEngine(HIGH);
override = true;
}
if(engineOn == true)
{
EngineKeyOnAcc();
delay(1000);
}
else if(engineOn == false)
{
EngineAcc();
delay(3000);
Starter();
delay(1000);
engineOn = true;
}
DefrostOff();
break;
case '7': //0111
if(override == false)
{
Override(HIGH);
OverrideEngine(HIGH);
override = true;
}
if(engineOn == true)
{
EngineKeyOnAcc();
}
else if(engineOn == false)
{
EngineAcc();
Starter();
engineOn = true;
}

DefrostOn();
break;
}
```

```
//5th character
//fan levels
if(engineOn == false || override == false)
{
    break;
}
delay(1000);
Serial.println("5th Character");
in_char = Serial1.read();
Serial.print(in_char);
switch(in_char)
{
    case '0': //0000
        setFan(0);
        break;
    case '1': //0001
        setFan(1);
        break;
    case '2': //0010
        setFan(2);
        break;
    case '4': //0100
        setFan(3);
        break;
    case '8': //1000
        setFan(4);
        break;
}
```

```
//6th character
//vent levels
delay(3000);
Serial.println("6th Character");
in_char = Serial1.read();
Serial.print(in_char);
switch(in_char)
{
    case '0': //0000
        setVent(0);
        break;
```

```
    case '1': //0001
      setVent(1);
      break;
    case '2': //0010
      setVent(2);
      break;
    case '4': //0100
      setVent(3);
      break;
    case '8': //1000
      setVent(4);
      break;
  }

  //7th character
  //temp
  delay(1000);
  Serial.println("7th Character");
  in_char = Serial1.read();
  Serial.print(in_char);
  switch(in_char)
  {
    case '1': //0001 hot
      setTemp(tempCold, 0);
      delay(10);
      setTemp(tempHot, 255);
      delay(10);
      break;
    case '2': //0010 cold
      setTemp(tempHot, 0);
      delay(10);
      setTemp(tempCold, 255);
      delay(10);
      break;
  }
  break;
}
}
```

```
}

//Send
if(checker == true)
{
  delay(2000);
  //testing
  Serial1.println("AT+CMGD=1,4"); // delete all SMS
  Serial.println();
  Serial.println("Sending");
  checker = false;
  //System off
  if(val == 0)
  {
    Serial.println("Sending System off");
  }
  //Ready to go
  else if(val == 1)
  {
    Serial.println("Sending System ready to go");
  }

  if(engineOn == false)
  {
    Serial.println("Engine not on");
  }
  else if(engineOn == true)
  {
    Serial.println("Engine on");
  }
  if(override == false)
  {
    Serial.println("Override not on");
  }
  else if(override == true)
  {
    Serial.println("Override on");
  }
}
```

```

Serial1.println("AT+CMGF=1"); // set SMS mode to text

Serial1.print("AT+CMGS="); // now send message...

Serial1.write(34); // ASCII equivalent of "

Serial1.print(mobilenumber);

Serial1.write(34);
Serial1.println();

delay(500); // give the module some thinking time
if(val == 0)
{
  Serial1.print("Off");
}
else
{
  Serial1.print("Ready");
}
Serial1.write(26); // ASCII equivalent of Ctrl-Z

delay(10000); // the SMS module needs time to return to OK status
checker = false; // stop sending messages, wait to receive

}
}

void Starter()
{
  //Set up for starter motor
  digitalWrite(enginePins[2], LOW); //Turn Engine A Off first - Radio/Cig_lighter
  digitalWrite(enginePins[4], HIGH); //Engine C - Starter Relay
  digitalWrite(enginePins[5], HIGH); //Engine D - Ignition Coil

  //Start the starter motor
  digitalWrite(enginePins[6], HIGH); //Engine E - Starter Motor

  delay(1500); //wait for 1.5 secs to turn and turn the starter motor off

```



```
digitalWrite(enginePins[6], LOW); //Turn off Engine E - Starter Motor
digitalWrite(enginePins[4], LOW); //Turn off Engine C - Starter Relay
digitalWrite(enginePins[2], HIGH); //Turn on Engine A - Radio/Cig_lighter
digitalWrite(enginePins[3], HIGH); //Turn on Engine B - Lights, Heat/Air, Misc.
}
```

```
void EngineKeyOnAcc()
```

```
{
digitalWrite(enginePins[6], LOW); //Turn off Engine E - Starter Motor
digitalWrite(enginePins[4], LOW); //Turn off Engine C - Starter Relay
digitalWrite(enginePins[5], HIGH); //Engine D - Ignition Coil
EngineAcc();
}
```

```
void EngineAcc()
```

```
{
//Turn on the accessories
digitalWrite(enginePins[2], HIGH); //Engine A - Radio/Cig_lighter
digitalWrite(enginePins[3], HIGH); //Engine B - Lights, Heat/Air, Misc.
//digitalWrite(enginePins[4], HIGH); //Engine C - Starter Relay
}
```

```
void StopEngine()
```

```
{
//Disconnect all pins except override
for(int i = 2; i < sizeof(enginePins); i++)
digitalWrite(enginePins[i], LOW);
}
```

```
void LockDoor()
```

```
{
digitalWrite(lockPins[1], HIGH);
delay(500); //wait for .5 secs to turn signal off
digitalWrite(lockPins[1], LOW);
}
```

```
void UnlockDoor()
```

```
{
digitalWrite(lockPins[2], HIGH);
delay(500); //wait for .5 secs to turn signal off
}
```

```
    digitalWrite(lockPins[2], LOW);
}
void ACOOn()
{
    digitalWrite(tempPins[1], HIGH);
}

void ACOff()
{
    digitalWrite(tempPins[1], LOW);
}
void DefrostOn()
{
    digitalWrite(defPins[0], HIGH);
}

void DefrostOff()
{
    digitalWrite(defPins[0], LOW);
}

void setFan(int lvl)
{
    digitalWrite(fanPins[1], HIGH);

    //Turn
    if(lvl == 0)
    {
        //Turn off every pins except Ovr
        digitalWrite(24, LOW);
        //delay(100);
        digitalWrite(26, LOW);
        //delay(100);
        digitalWrite(28, LOW);
        //delay(100);
        digitalWrite(30, LOW);
    }
    else
    {
        digitalWrite(26, LOW);
    }
}
```

```

//delay(100);
digitalWrite(28, LOW);
//delay(100);
digitalWrite(30, LOW);

if(lvl > 1) {
    digitalWrite(fanPins[lvl], HIGH);
}
}
}

void setTemp(int address, int value)
{
    // take the SS pin low to select the chip:
    digitalWrite(slaveSelectPin,LOW);
    // send in the address and value via SPI:
    SPI.transfer(address);
    SPI.transfer(value);
    // take the SS pin high to de-select the chip:
    digitalWrite(slaveSelectPin,HIGH);
}

void setVent(int lvl)
{
    //Disconnect every pin except Ovr
    //for(int i = 1; i < sizeof(ventPins); i++)
    //digitalWrite(ventPins[i], LOW);
    digitalWrite(ventPins[1], LOW);
    digitalWrite(ventPins[2], LOW);
    digitalWrite(ventPins[3], LOW);
    digitalWrite(ventPins[4], LOW);
    digitalWrite(ventPins[5], LOW);

    //if lvl=0 -> A; lvl=1 -> B; lvl=2 -> C; lvl=3 -> D; lvl=4 -> E;
    digitalWrite(ventPins[lvl+1], HIGH);
}

//Adjust all override signals
void Override(int OnOff)
{

```

```

//OnOff = LOW or HIGH
digitalWrite(fanPins[0], OnOff);
//digitalWrite(enginePins[0], OnOff);
//digitalWrite(enginePins[1], OnOff);
digitalWrite(ventPins[0], OnOff);
//digitalWrite(lockPins[0], OnOff);
//digitalWrite(defPins[0], OnOff);
digitalWrite(tempPins[0], OnOff);
}

//Adjust all override signals
void OverrideEngine(int OnOff)
{
//OnOff = LOW or HIGH
digitalWrite(enginePins[0], OnOff);
digitalWrite(enginePins[1], OnOff);
}

//Reset all pins
void ResetAll()
{
//Set pins to output and initialize to low
for(int i = 0; i < sizeof(fanPins); i++)
{
pinMode(fanPins[i], OUTPUT);
digitalWrite(fanPins[i], LOW);
}
for(int i = 0; i < sizeof(enginePins); i++)
{
pinMode(enginePins[i], OUTPUT);
digitalWrite(enginePins[i], LOW);
}
for(int i = 0; i < sizeof(ventPins); i++)
{
pinMode(ventPins[i], OUTPUT);
digitalWrite(ventPins[i], LOW);
}
for(int i = 0; i < sizeof(lockPins); i++)
{
pinMode(lockPins[i], OUTPUT);
}
}

```

```
    digitalWrite(lockPins[i], LOW);
}
for(int i = 0; i < sizeof(defPins); i++)
{
    pinMode(defPins[i], OUTPUT);
    digitalWrite(defPins[i], LOW);
}
for(int i = 0; i < sizeof(tempPins); i++)
{
    pinMode(tempPins[i], OUTPUT);
    digitalWrite(tempPins[i], LOW);
}
}
```

## Android Code

**Project.java** (the main logic)

```
package senior.project;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
//import android.graphics.Color;
import android.os.Bundle;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.SeekBar;
//import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.telephony.SmsManager;
import android.view.View;
import android.view.View.OnClickListener;

public class Project extends Activity implements OnClickListener {
```

```
/** Called when the activity is first created. */
String number;
static boolean overOn;

static Button send;
static ProgressBar progress;

static TextView displayTemp;
static TextView displayFan;
static TextView displayVent;
static TextView displayLock;

static SeekBar seekTemp;
static SeekBar seekFan;
static SeekBar seekVent;
static SeekBar seekLock;

static ToggleButton engine;
static ToggleButton ac;
static ToggleButton defrost;
static ToggleButton override;

static boolean arduinoReady;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    number = "8015606224";
    //number = "5556";
    overOn = false;

    send = (Button) findViewById(R.id.button1);
    // progress = (ProgressBar)findViewById(R.id.progressBar1);

    displayTemp = (TextView) findViewById(R.id.textView6);
    displayFan = (TextView) findViewById(R.id.textView8);
    displayVent = (TextView) findViewById(R.id.textView10);
    displayLock = (TextView) findViewById(R.id.textView12);
```

```
seekTemp = (SeekBar) findViewById(R.id.seekBar1);
seekFan = (SeekBar) findViewById(R.id.seekBar2);
seekVent = (SeekBar) findViewById(R.id.seekBar3);
seekLock = (SeekBar) findViewById(R.id.seekBar4);

override = (ToggleButton) findViewById(R.id.toggleButton1);
engine = (ToggleButton) findViewById(R.id.toggleButton2);
ac = (ToggleButton) findViewById(R.id.toggleButton3);
defrost = (ToggleButton) findViewById(R.id.toggleButton4);

seekTemp.setMax(1);
seekFan.setMax(4);
seekVent.setMax(4);
seekLock.setMax(2);

//Set temp seekbar to start at position 3
//seekTemp.setProgress(3);
//displayTemp.setText("Normal");

// //////////////////////////////////////
// State of the car. May change to having the car send the state to the
// phone on start of the app.
// before the phone can do anything
engine.setEnabled(false);
ac.setEnabled(false);
defrost.setEnabled(false);
//override.setEnabled(false);
seekTemp.setEnabled(false);
seekFan.setEnabled(false);
seekVent.setEnabled(false);
//seekLock.setEnabled(false);
// //////////////////////////////////////

send.setOnClickListener(this);
engine.setOnClickListener(this);
ac.setOnClickListener(this);
defrost.setOnClickListener(this);
override.setOnClickListener(this);

seekTemp.setOnSeekBarChangeListener(seekBarChangeListener);
```

```

        seekFan.setOnSeekBarChangeListener(seekBarChangeListener);
        seekVent.setOnSeekBarChangeListener(seekBarChangeListener);
        seekLock.setOnSeekBarChangeListener(seekBarChangeListener);
    }

    private SeekBar.OnSeekBarChangeListener seekBarChangeListener = new
SeekBar.OnSeekBarChangeListener() {

        public void onProgressChanged(SeekBar seekBar, int progress,
            boolean fromUser) {
            // TODO Auto-generated method stub
            if (overOn) {
                updateBar();
                updateTemp();
                updateVent();
            }
            updateLock();
            /*else {
                updateZero();
            }*/
        }

        public void onStartTrackingTouch(SeekBar seekBar) {
            // TODO Auto-generated method stub
        }

        public void onStopTrackingTouch(SeekBar seekBar) {
            // TODO Auto-generated method stub
        }
    };

    private void updateBar() {
        displayTemp.setText(String.valueOf(seekTemp.getProgress()));
        displayFan.setText(String.valueOf(seekFan.getProgress()));
        displayVent.setText(String.valueOf(seekVent.getProgress()));
        displayLock.setText(String.valueOf(seekLock.getProgress()));
    }

    private void updateZero() {
        //displayTemp.setText(String.valueOf(0));
        displayTemp.setText("Cold");
    }

```



```

        displayFan.setText(String.valueOf(0));
        displayVent.setText("Face");
        displayLock.setText("Nothing");
    }
    private void updateLock(){
        if(seekLock.getProgress() == 0)
            displayLock.setText("Nothing");
        else if(seekLock.getProgress() == 1)
            displayLock.setText("Lock");
        else if(seekLock.getProgress() == 2)
            displayLock.setText("Unlock");
    }
    private void updateVent(){
        if(seekVent.getProgress() == 0)
            displayVent.setText("Defrost");
        else if(seekVent.getProgress() == 1)
            displayVent.setText("FD");
        else if(seekVent.getProgress() == 2)
            displayVent.setText("Foot");
        else if(seekVent.getProgress() == 3)
            displayVent.setText("FF");
        else if(seekVent.getProgress() == 4)
            displayVent.setText("Face");
    }
    private void updateTemp(){
        if(seekTemp.getProgress() == 0)
        {
            displayTemp.setText("Cold");
            ac.setEnabled(true);
        }
        else if(seekTemp.getProgress() == 1)
        {
            displayTemp.setText("Hot");
            ac.setChecked(false);
            ac.setEnabled(false);
        }
    }
}

public void onClick(View src) {
    // TODO Auto-generated method stub
    String message = "";

```

```

switch (src.getId()) {
    case R.id.toggleButton1:
        if (override.isChecked() == false) {
            overOn = false;

            engine.setEnabled(false);
            ac.setEnabled(false);
            defrost.setEnabled(false);

            seekTemp.setEnabled(false);
            seekFan.setEnabled(false);
            seekVent.setEnabled(false);
            //seekLock.setEnabled(false);

            updateZero();
        } else if (override.isChecked() == true) {
            overOn = true;

            engine.setEnabled(true);
            ac.setEnabled(true);
            defrost.setEnabled(true);

            seekTemp.setEnabled(true);
            seekFan.setEnabled(true);
            seekVent.setEnabled(true);
            //seekLock.setEnabled(true);

            updateBar();
            updateTemp();
            updateVent();
            updateLock();
        }
        break;

    case R.id.button1:
        message = "zz";
        // Override Off
        if (override.isChecked() == false)
        {
            //Nothing

```

```

if(seekLock.getProgress() == 0)
{
    message += "00000";
}
//Lock
else if(seekLock.getProgress() == 1)
{
    message += "80000";
}
//Unlock
else if(seekLock.getProgress() == 2)
{
    message += "40000";
}
}
// Override On
else {
    //3rd character
    //AC, Lock, Unlock
    //AC on
    if (ac.isChecked())
    {
        //Nothing
        if(seekLock.getProgress() == 0)
        {
            message += "2";
        }
        //Lock
        else if(seekLock.getProgress() == 1)
        {
            message += "a";
        }
        //Unlock
        else if(seekLock.getProgress() == 2)
        {
            message += "6";
        }
    }
    //AC on
    else if (ac.isChecked() == false)

```

```
{
    //Nothing
    if(seekLock.getProgress() == 0)
    {
        message += "0";
    }
    //Lock
    else if(seekLock.getProgress() == 1)
    {
        message += "8";
    }
    //Unlock
    else if(seekLock.getProgress() == 2)
    {
        message += "4";
    }
}
```

```
//4th character
//Override, Engine, Defrost
if(engine.isChecked())
{
    // Defrost
    if (defrost.isChecked())
        message += "7";
    else if (defrost.isChecked() == false)
        message += "3";
}
else if(engine.isChecked() == false)
{
    message += "1";
}
```

```
//5th character
//Fan
if(seekFan.getProgress() == 0)
{
    message += "0";
}
else if(seekFan.getProgress() == 1)
{
```

```
        message += "1";
    }
    else if(seekFan.getProgress() == 2)
    {
        message += "2";
    }
    else if(seekFan.getProgress() == 3)
    {
        message += "4";
    }
    else if(seekFan.getProgress() == 4)
    {
        message += "8";
    }

//6th character
//Vent
if(seekVent.getProgress() == 0)
{
    message += "0";
}
else if(seekVent.getProgress() == 1)
{
    message += "1";
}
else if(seekVent.getProgress() == 2)
{
    message += "2";
}
else if(seekVent.getProgress() == 3)
{
    message += "4";
}
else if(seekVent.getProgress() == 4)
{
    message += "8";
}

//7th character
//Temp
//Cold
```

```

        if(seekTemp.getProgress() == 0)
        {
            message += "1";
        }
        //Hot
        else if(seekTemp.getProgress() == 1)
        {
            message += "2";
        }
    }

    engine.setEnabled(false);
    ac.setEnabled(false);
    defrost.setEnabled(false);
    override.setEnabled(false);
    seekTemp.setEnabled(false);
    seekFan.setEnabled(false);
    seekVent.setEnabled(false);
    seekLock.setEnabled(false);
    send.setEnabled(false);

    sendSMS(number, message);
    break;
}
}

public static void off ()
{
    engine.setEnabled(false);
    ac.setEnabled(false);
    defrost.setEnabled(false);
    override.setEnabled(false);
    seekTemp.setEnabled(false);
    seekFan.setEnabled(false);
    seekVent.setEnabled(false);
    seekLock.setEnabled(false);
    send.setEnabled(false);
}

public static void update ()
{

```

```

override.setEnabled(true);
seekLock.setEnabled(true);
send.setEnabled(true);
if (override.isChecked() == false) {
    overOn = false;

    engine.setEnabled(false);
    ac.setEnabled(false);
    defrost.setEnabled(false);

    seekTemp.setEnabled(false);
    seekFan.setEnabled(false);
    seekVent.setEnabled(false);
    //seekLock.setEnabled(false);

} else if (override.isChecked() == true) {
    overOn = true;

    engine.setEnabled(true);
    ac.setEnabled(true);
    defrost.setEnabled(true);

    seekTemp.setEnabled(true);
    seekFan.setEnabled(true);
    seekVent.setEnabled(true);
    //seekLock.setEnabled(true);
}
}

// ---sends an SMS message to another device---
private void sendSMS(String phoneNumber, String message) {
    String SENT = "SMS_SENT";
    String DELIVERED = "SMS_DELIVERED";

    PendingIntent sentPI = PendingIntent.getBroadcast(this, 0, new Intent(
        SENT), 0);

    PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0,
        new Intent(DELIVERED), 0);

    // ---when the SMS has been sent---

```

```

registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context arg0, Intent arg1) {
        switch (getResultCode()) {
            case Activity.RESULT_OK:
                Toast.makeText(getBaseContext(), "SMS sent",
                    Toast.LENGTH_SHORT).show();
                break;
            case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                Toast.makeText(getBaseContext(), "Generic failure",
                    Toast.LENGTH_SHORT).show();
                break;
            case SmsManager.RESULT_ERROR_NO_SERVICE:
                Toast.makeText(getBaseContext(), "No service",
                    Toast.LENGTH_SHORT).show();
                break;
            case SmsManager.RESULT_ERROR_NULL_PDU:
                Toast.makeText(getBaseContext(), "Null PDU",
                    Toast.LENGTH_SHORT).show();
                break;
            case SmsManager.RESULT_ERROR_RADIO_OFF:
                Toast.makeText(getBaseContext(), "Radio off",
                    Toast.LENGTH_SHORT).show();
                break;
        }
    }
}, new IntentFilter(SENT));

```

```

// ---when the SMS has been delivered---
registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context arg0, Intent arg1) {
        switch (getResultCode()) {
            case Activity.RESULT_OK:
                Toast.makeText(getBaseContext(), "SMS delivered",
                    Toast.LENGTH_SHORT).show();
                break;
            case Activity.RESULT_CANCELED:
                Toast.makeText(getBaseContext(), "SMS not
delivered",
                    Toast.LENGTH_SHORT).show();

```



```

                break;
            }
        }
    }, new IntentFilter(DELIVERED));

    SmsManager sms = SmsManager.getDefault();
    sms.sendTextMessage(phoneNumber, null, message, sentPI, deliveredPI);
}
}

```

### **SmsReceiver.java** (message receive logic)

```

package senior.project;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;

public class SmsReceiver extends BroadcastReceiver {
    String message = "";
    static String mes = "";

    @Override
    public void onReceive(Context context, Intent intent) {
        // ---get the SMS message passed in---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "";
        if (bundle != null) {
            // ---retrieve the SMS message received---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];

            for (int i = 0; i < msgs.length; i++) {
                msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
                str += "SMS from " + msgs[i].getOriginatingAddress();
                str += " :";
                str += msgs[i].getMessageBody().toString();
                message += msgs[i].getMessageBody().toString();
            }
        }
    }
}

```

```

        str += "\n";
    }

    mes = message;
    if(message.equals("Ready"))
    {
        Project.update();
    }
    else if(message.equals("Off"))
    {
        Project.off();
    }
    // ---display the new SMS message---
    Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
}
}

public static String getMessage() {
    return mes;
}
}

```

### **AndroidManifest.xml** (the config)

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="senior.project"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />

    <uses-permission android:name="android.permission.SEND_SMS">
    </uses-permission>
    <uses-permission android:name="android.permission.RECEIVE_SMS">
    </uses-permission>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".Project"

```

```

        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <receiver android:name=".SmsReceiver">
        <intent-filter>
            <action android:name=
                "android.provider.Telephony.SMS_RECEIVED" />
        </intent-filter>
    </receiver>

</application>
</manifest>

```

### main.xml (the layout)

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center" >

            <ToggleButton
                android:id="@+id/toggleButton1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="ToggleButton" />

            <ToggleButton

```

```
android:id="@+id/toggleButton2"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="ToggleButton" />
```

```
<ToggleButton  
android:id="@+id/toggleButton3"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="ToggleButton" />
```

```
<ToggleButton  
android:id="@+id/toggleButton4"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="ToggleButton" />
```

```
</TableRow>
```

```
<TableRow  
android:id="@+id/tableRow2"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:gravity="center" >
```

```
<TextView  
android:id="@+id/textView1"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Override"  
android:textAppearance="?android:attr/textAppearanceSmall"  
android:layout_gravity="center"/>
```

```
<TextView  
android:id="@+id/textView2"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Engine"  
android:textAppearance="?android:attr/textAppearanceSmall"  
android:layout_gravity="center"/>
```

```
<TextView
```

```
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="AC"
        android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_gravity="center"/>
```

```
        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Defrost"
            android:textAppearance="?android:attr/textAppearanceSmall"
        android:layout_gravity="center"/>
    </TableRow>
```

```
    <TableRow
        android:id="@+id/tableRow11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
    </TableRow>
```

```
</TableLayout>
```

```
    <TableRow
        android:id="@+id/tableRow4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
```

```
        <TextView
            android:id="@+id/textView0"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=" "
            android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
</TableRow>
```

```
    <RelativeLayout
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content" android:gravity="center"  
android:layout_gravity="center">
```

```
</RelativeLayout>
```

```
<TableLayout
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" android:layout_gravity="center">
```

```
<TableRow
```

```
    android:id="@+id/tableRow3"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:gravity="center" >
```

```
<TextView
```

```
    android:id="@+id/textView5"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Temp"  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
<SeekBar
```

```
    android:id="@+id/seekBar1"  
    android:layout_width="wrap_content"  
    android:layout_height="match_parent" />
```

```
<TextView
```

```
    android:id="@+id/textView6"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="right"  
    android:layout_marginLeft="5px"  
    android:gravity="right"  
    android:text="Cold"  
    android:textAppearance="?android:attr/textAppearanceMedium"/>
```

```
</TableRow>
```

```
<TableRow
```

```
    android:id="@+id/tableRow9"  
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content" android:minHeight="30px">
</TableRow>
```

```
<TableRow
    android:id="@+id/tableRow5"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center" >
```

```
<TextView
    android:id="@+id/textView7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Fan"
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
<SeekBar
    android:id="@+id/seekBar2"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

```
<TextView
    android:id="@+id/textView8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:layout_marginLeft="5px"
    android:gravity="right"
    android:text="0"
    android:textAppearance="?android:attr/textAppearanceMedium"/>
</TableRow>
```

```
<TableRow
    android:id="@+id/tableRow10"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:minHeight="30px">
</TableRow>
```

```
<TableRow
    android:id="@+id/tableRow6"
    android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"  
android:gravity="center" >
```

```
<TextView  
    android:id="@+id/textView9"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Vent"  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
<SeekBar  
    android:id="@+id/seekBar3"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

```
<TextView  
    android:id="@+id/textView10"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="right"  
    android:layout_marginLeft="5px"  
    android:gravity="right"  
    android:text="Face"  
    android:textAppearance="?android:attr/textAppearanceMedium"/>
```

```
</TableRow>
```

```
<TableRow  
    android:id="@+id/tableRow8"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:gravity="center"  
    android:visibility="invisible" >
```

```
<TextView  
    android:id="@+id/textView11"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Space "  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
<TextView
```



```
    android:id="@+id/textView13"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Medium"  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
</TableRow>
```

```
<TableRow
```

```
    android:id="@+id/tableRow12"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" android:gravity="center">
```

```
<TextView
```

```
    android:id="@+id/textView14"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Lock"  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
<SeekBar
```

```
    android:id="@+id/seekBar4"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

```
<TextView
```

```
    android:id="@+id/textView12"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Nothing"  
    android:textAppearance="?android:attr/textAppearanceMedium"  
    android:gravity="right" android:minWidth="100px"/>
```

```
</TableRow>
```

```
</TableLayout>
```

```
<TableRow
```

```
    android:id="@+id/tableRow7"  
    android:layout_width="match_parent"
```

```
        android:layout_height="match_parent" android:layout_gravity="bottom|center"  
        android:gravity="bottom|center">
```

```
    <Button  
        android:id="@+id/button1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Send" />
```

```
</TableRow>
```

```
</LinearLayout>
```