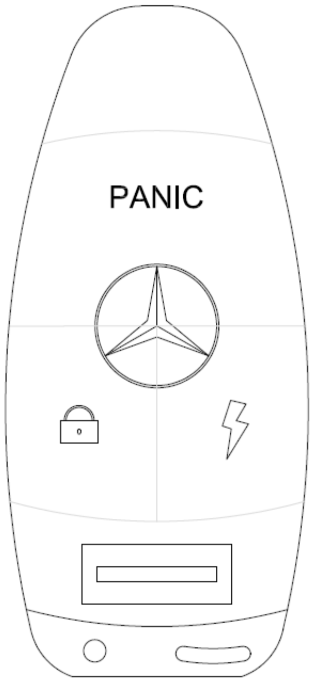
SMART – AKK



The SMART-AKK is a small radio transmitter which grants remote access to your vehicle using a biometric verification module and a Feistel encryption scheme.

April 12, 2010

Anita Has ahas@ualberta.ca
Kristina Suen suen@ualberta.ca

Declaration of Original Content

"The design elements of this project and report are entirely the original work of the authors and have not been submitted for credit in any other course except as follows:"

- Suggestion of using a level transceiver when interfacing the biometric module was by Dr. Duncan Elliot
- Format on how to connect RS232 was referenced from: http://www.ece.ualberta.ca/~cmpe401/fall2004/labs/lab2/CMPE401Fall2004Lab2.pdf
- Block diagram of connecting the fingerprint module to the ARM via RS232 was originally drawn by Nancy Minderman.

____________________________	____________________________
Anita Has	Kristina Suen

I. ABSTRACT

The SMART-AKK is aimed at automotive manufacturers looking for enhanced security in today's SmartKeys. The SMART-AKK is a small radio transmitter which grants remote access to a user's vehicle. The need for this product stems from the security vulnerability involved with a lost or stolen SmartKey and eavesdropping on a line of communication. If a professional auto thief is able to eavesdrop on the line and replicate the signals from transmitter to receiver, there is nothing stopping them from driving away with your vehicle.

The SMART-AKK was designed to lower those security vulnerabilities involved in lost or stolen keys as wells as eavesdropping. Our design lowered those security vulnerabilities by using cutting edge biometric technologies and encryption of a synchronized serial number. This is done by using a combination of Tiny Encryption Algorithm (TEA) block cipher over the wireless communication line and an ARA-ME-01 biometric fingerprint scanner. SMART-AKKs main feature, the biometric fingerprint scanner introduces a new layer of protection by incorporating different levels of access for the vehicle.

To demonstrate the proper functionality of the SMART-AKK, a vehicle simulator (AKV) was included in the design. The AKV communicates with SMART-AKK via a wireless link using two X-Bee wireless modules.

SMART-AKK, and the vehicle simulator and all features have been tested individually as well as collectively and have produced expected results. This document will carry out a design review of the requirements and specifications.

II. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

ABBREVIATION	DEFINITION
AKV	Vehicle Simulator
TEA	Tiny Encryption Algorithm
Atmel55	AT91EB55 Microchip

TABLE OF CONTENTS

1 FUNCTIONAL REQUIREMENTS

SMART - AKK is required to grant a valid user remote access to a vehicle through the use of a biometric verification module that is capable of adding/removing and verifying at least 40 human fingerprints. The remote access is required to be via a wireless link with a minimum range of 10 meters. AKK must significantly decrease security vulnerability caused by eavesdropping through the implementation of a block cipher and synchronized serial counter.

The AKK has three push buttons to allow the user the following functionality:

I. Lock/Unlock All Doors
II. Engine On/Off
III. Panic

The ARA-ME biometric module that was chosen for SMART-AKK is capable of storing 120 finger prints, including one master fingerprint. It is able to perform all finger print scanning and matching on chip in less than 3 seconds. SMART-AKK is able to use ARA-ME to enroll more fingerprints, remove all stored fingerprint and reset a master fingerprint by pressing a sequence of buttons.

The wireless module chosen for SMART-AKK was the X-Bee. It is able to transmit and receive 32-bit packets with a range of at least 10 meters. If the SMART-AKK and AKV are out of range of each other, the SMART-AKK will fail on transmitting packets and AKV will not execute any command. Additionally, if packets were not received correctly or if packets were lost on the AKV side, AKV will simply treat the received packet as an invalid command and not execute any action.

To guarantee a decrease in security vulnerability by eavesdroppers, both SMART-AKK and AKV use the Tiny Encryption Algorithm on 64-bit packets with a 128-bit key. This block cipher was chosen for its simplicity in implementation, speed and cryptographic strength. It achieves complete diffusion after only six rounds and therefore is a strong encryption format for SMART-AKK.

The synchronized serial counter is another way SMART-AKK reduces eavesdropping. Both the AKV and the SMART-AKK have serial number counters that increment every time a packet is sent or received. If AKV receives a packet with a lower serial number than its counter, it will discard the packet and treat it as invalid. It will not execute a command if the same packet was sent twice.

2 DESIGN & DESCRIPTION OF OPERATION

SMART-AKK uses LEDs on its development board as verification indicators. This can be seen in Table 2.1 below.

LED NUMBER	LED STATE	
	ON	BLIKING
LED1	fingerprint valid	finger print invalid
LED2	button sequence valid	button sequence invalid
LED3	encryption completed successfully	encryption failed
LED4	sent packet successfully	packet sending failed
LED8	timeout	ready

Table 2.1: SMART-AKK LED Indicator's

A user will commence interaction with the SMART-AKK by scanning their fingerprint on the ARA-ME module for authentication. The module will then verify the validity of the presented fingerprint.

A user will then press one of 3 buttons in a sequence on the SMART-AKK that they wish AKV to execute (Table 2.2 and Table 2.3 below show valid biometric and button sequences for the AKV, respectively). ARA-ME-01 will process the fingerprint and Atmel55 will verify and validate the button press.

Valid ARA-ME Commands	Atmel55 Button Sequence
Enroll Finger Prints	Button4 – Button3 – Button2 – Button1
Remove Finger Prints & Enroll Master	Button3 Button4 Button3 Button4

Table 2.2: Biometric Command Request

Valid AKV Commands	Atmel55 Button Sequence	32-bit Action
Unlock All Doors	Button 1	0x11111111
Lock All Doors	double click Button 1	0x11112222
Engine On	Button 2	0x22222222
Engine Off	double click Button 2	0x22223333
Panic	Button 3	0x33333333

Table 2.3: Valid AKV Commands with respect to SMART-AKK Button Sequences

SMART-AKK will increment the 32-bit serial number counter by a hexadecimal value of 0x8 and append it with the 32-bit AKV Command (as seen in Figure 2.1 below), creating a 64-bit packet. The 64-bit packet is then encrypted using TEA.

The packet will be appended with a start byte of 0xAA and an end byte of 0xFF. These will act as a signal for a transmission request to AKV.

0xAA	AKV Action	Serial Number	0xFF

Figure 2.1: SMART-AKK Packet Format

Once the packet has been received by the X-Bee wireless module on the AKV will first decrypt the packet. AKV will then compare the SMART-AKK Serial Number. If it is less than its local serial number AKV does nothing (in the possibility that an eavesdropper replicated the signal). Otherwise, it increments the local serial count by a hexadecimal number of 0x8. Then it proceeds to check if the requested AKV Action was valid.

If AKV received a valid command it will execute the respective command with the following LED sequences as listed in Table 2.4.

Requested AKV Command	AKV Execution
Unlock All Doors	MASK blink once
Lock All Doors	MASK blink twice
Engine On	MASK light sequentially from left to right, three times
Engine Off	MASK light sequentially from right to left, once
Panic	MASK blink four times

Table 2.4: AKV LED Execution Patterns

The platform chosen for both SMART-AKK and AKV is the Atmel AT91E55 Single Board Computer

(Atmel55). Various modules will be connected and interfaced with the board to implement specific features of SMART-AKK as shown in Figure 2.2 below.

To implement the biometric authentication of the SMART-AKK, the ARA-ME-01 will be used. Authentication of fingerprints is done on the module's microchip. It has the ability to store 120 fingerprints and perform fingerprint scanning and matching is less than 3 seconds. To communicate to the biometric module, a series of commands and responses will be sent and received from the module in hexadecimal format. Please refer to the software design section for a more detailed description.

Communication between the SMART-AKK and AKV will be done wirelessly using the X-Bee ZB embedded RF module. Two modules will be required; one mounted on the AKK and the other mounted on the AKV.

Both ARA-ME-01 and X-Bee are connected to the Atmel55 development board via USART1 and USART2 respectively. It is to be noted that ARA-ME-01 is connected to USART1 via an RS232 level transceiver chip to account for voltage differences between the Atmel55 serial port and the module and possible contention issues with connecting directly to USART1.

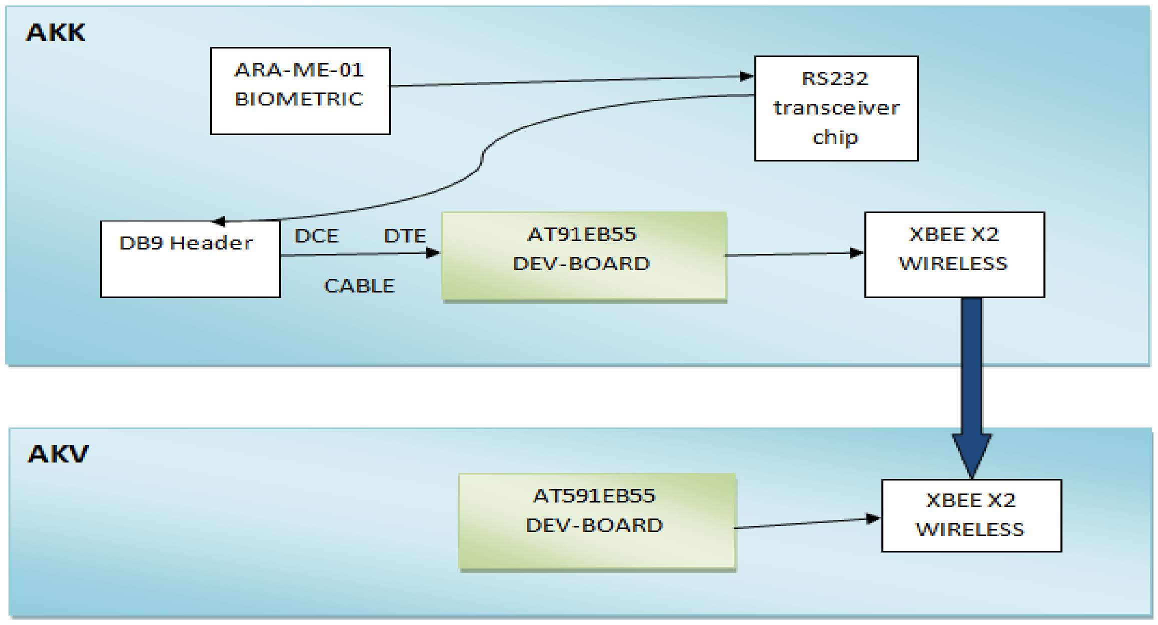


Figure 2.2: Hardware Layout

Please refer to Appendix III for a more detailed design of the hardware connections.

3 PARTS LIST

Part Name	SKU Number	Supplier	Cost CDN $	Documentation
Fingerprint Slide Scanner ARA-ME-01	SEN-08881	Sparkfun	79.95	Specification
Fingerprint Slide Scanner Interface Cable	SEN-08941	Sparkfun	1.95	N/A
XBee 1mW Chip Antenna (x2)	WRL-08664	Sparkfun	22.95	Data Sheet
Atmel AT91E55 Single Board Computer (x2)	N/A	ECE Parts Store	N/A	N/A
RS232 Converter DIP - MAX232 MAX3232	COM-00316	Sparkfun	1.95	Data Sheet
1 uF Capacitor (x4) ECA-1HM010I	P10421TB-ND	DigiKey	0.03	N/A
DB9 Female Serial Header	N/A	ECE Parts Store	N/A	N/A

Table 3.1: SMART-AKK hardware parts

Note: The XBee development kit contains RS-232 interface boards.

Detailed hardware specification for SMART-AKK can be seen in the table below:

Part Name	Supply Voltage [V]	Supply Current [mA]	Interface	Other
ARA-ME-01	5	60	9600bps TTL Serial 3.3V CMOS UART 3.3V CMOS DIGITAL I/O	Resolution: 500 DPI
X-Bee	3.3	50	250kbps Serial 3.3V CMOS UART	2.4 GHz Range: 100 [m] 128-bit encryption 6 10-bit ADC input pins
MAX232	2.7 - 6	0.3	Minimum 120kbps Serial	Output Resistance: 300 Ω Requires 4 external capacitors

Table 3.2: detailed SMART-AKK hardware parts

4 DATA SHEET

ARA-ME-01			Connection on ARM		
Name	Pin	Input / Output	Function	Name	Pin
GND	1	-	Ground	-	-
Reset	2	Input	Module reset control	PA23	95 (16 on JPCOMM)
TXD	3	Output	Serial sender	PA19/RXD1 *	91 (9 on JPCOMM)
RXD	4	Input	Serial receiver	PA18/TXD1 *	86 (10 on JPCOMM)
Vcc	6	-	Power supply 5V	-	-

Table 4.1: connections of the fingerprint slide scanner

∗These are the final pins that the ARA-ME-01 module is connected to. It does however connect to these pins through a level transceiver. Please refer to Appendix III (3/4) for more details.

X-Bee			Connection on ARM		
Name	Pin	Input / Output	Function	Name	Pin
Vcc	1	-	Power supply 3.3V	-	-
DOUT	2	Output	UART Data out	PA22/RXD2	94 (14 on JPCOMM)
DIN/CONFIG	3	Input	UART Data in	PA21/TXD2	93 (11 on JPCOMM)
GND	10	-	Ground	-	-

Table 4.2: connections of the X-Bee Module

X-Bee configuration settings

Setting	SMART-AKK	AKV
Baud rate	38400	38400
PAN ID	0x6666	0x6666
DL	0	1
MY	1	0

Table 4.3: X-Bee configuration settings

Note: X-Bee modules are initialized using the development kit. Please refer to Appendix V on how to configure the X-Bee modules.

For both the AKK and AKV, the X-Bee module is connected to UART2 of the Atmel board. Commands are passed to the X-Bee for transmission and retrieving data using serial at91 commands.

- Switch buttons

  To accommodate the functionality of the AKK, the four switches (available directly on the Atmel55) will be used. Button presses are detected using polling and by checking the status of the PIOA (for button 2) and PIOB (for button1, 3 and 4) bus using at91_pio_read.

Push button	ARM PIN
Button1	PB20
Button2	PA9
Button3	PB17
Button4	PB19

Table 4.4: pin numbers of push buttons on the ARM board

- LEDs

  To display the action requested by a user on the AKV, LEDs on the Atmel55 will be used to display the requested functionality as described in our functional requirements. LEDs are turned on/off using the at91_pio_open and at91_pio_write functions.

LED	Pin on ARM
LED1	PB8
LED2	PB9
LED3	PB10
LED4	PB11
LED5	PB12
LED6	PB13
LED7	PB14
LED8	PB15

Table 4.5: pin numbers of LEDs on the ARM board

- Power Draw
    - Biometric module

Measurement	Theoretical	Measured	Unit
Idle voltage (module powered but not working)	5	4.9	V
Working voltage (ie. When biometric module is receiving/sending commands)	5	4.93	V
Idle current	10	6.8	mA
Working current	60	56.8	mA
Working power (P=IV)	0.3	0.28	W
Idle power (P=IV)	0.05	0.033	W

Table 4.6: power measurements of ARA-ME-01 biometric module

Working voltage and current was measured when commands and responses were being sent and received to and from the biometric module. Idle voltage and current were measured when the biometric module was powered and running, however, however no commands or responses were being sent or received.

- X-Bee

Measurement	Theoretical	Measured	Unit
Voltage (when idle, transmitting and receiving)	3.1	3.061	V
Transmit current	35	20.3	mA
Receive current	38	24.3	mA
Transmit power (P=IV)	0.1085	0.062	W
Receive power (P=IV)	0.1178	0.074	W

Table 4.7: power measurements of the X-Bee module

Transmit current was measured on the SMART-AKK X-Bee when transmitting packets to AKV. Similarly, receive current was measured on the AKV X-Bee when receiving packets from SMART-AKK.

5 SOFTWARE DESIGN

The software components are divided in two sections; software relating to SMART-AKK and software relating to the AKV. The software flow of the SMART-AKK and AKV can be seen in Figure 5.1 and 5.2 below. Please refer to Appendix IV for a block diagram of software interactions.

Note: The solid dark blue arrows signify valid or normal flow.

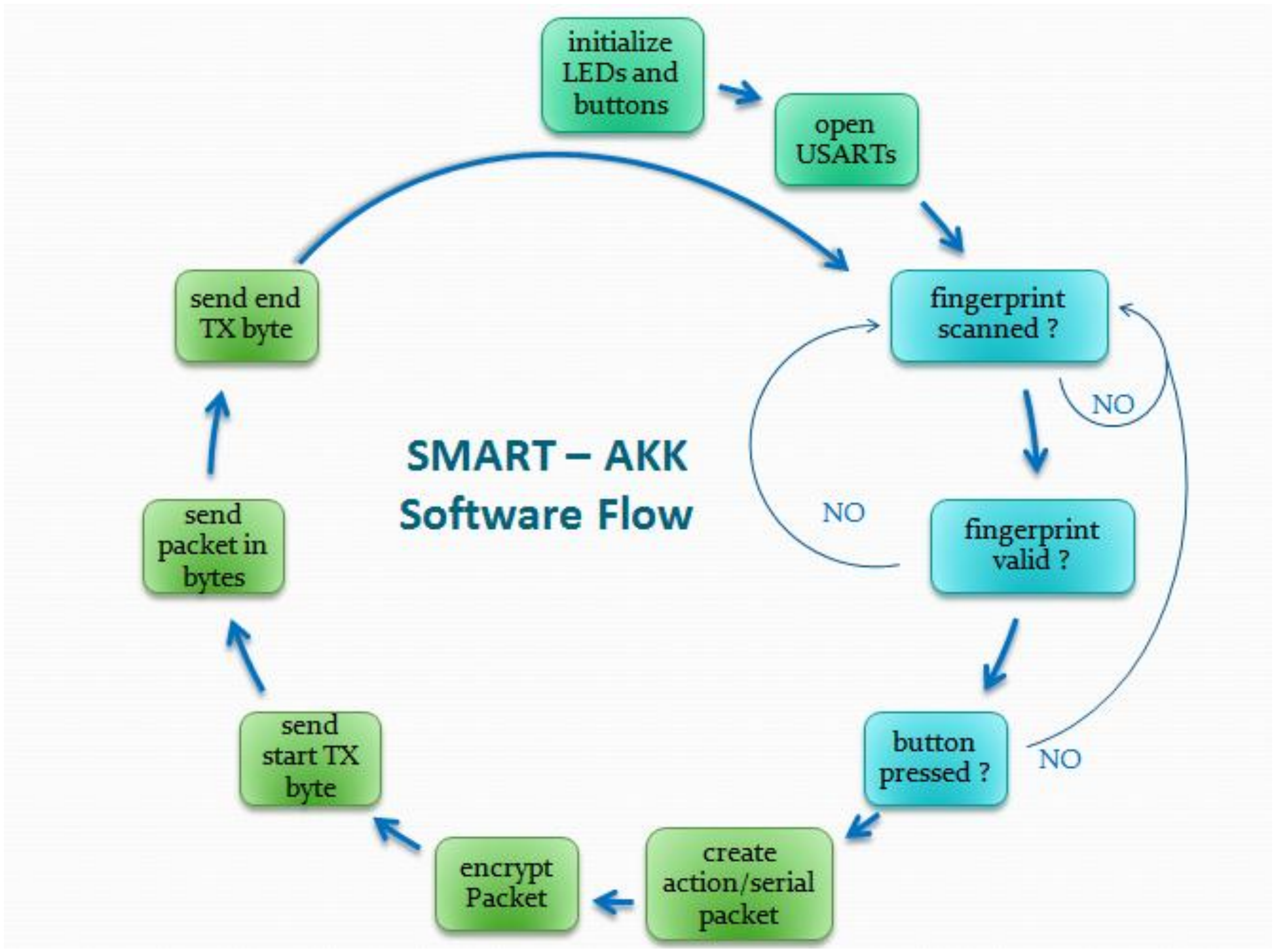


Figure 5.1: SMART-AKK Software Data Flow

SMART-AKK will be retrieving data from the fingerprint scanner and button presses, generating a code and serial number, encrypting data packets, and sending them via wireless module. SMART-AKK first initializes the LEDs and buttons. Then it opens the USARTs necessary to communicate with the external modules (USART2 and USART1 for X-Bee and the fingerprint module, respectively). If a fingerprint was scanned, it will validate the fingerprint and check if a button was pressed. If any of these actions fail, it will go back and poll for another fingerprint. Otherwise, it will create and encrypt an action/serial packet. Then, it will begin sending the packets to AKV starting with a start transmit byte, 0xAA, and ending with an end transmit byte, 0xFF. Once this has finished, SMART-AKK will begin polling for a new fingerprint yet again.

To communicate with the biometric module Atmel55 must send and receive a sequence of commands and replies in hexadecimal format. Table 5.1 below, is an example packet to read an image from the sensor, and the associated response that one would receive from the module. The response shown is a standard packet that the module sends when a command was executed successfully.

Brief Summary of Packet
The packet head and device address is the same for the request and response. The 01 in the packet flag denotes a command packet and 07 denotes a reply from the module. The packet length is the length in bytes of the command and checksum. Checksum is the sum of the packet flag, length and command. There is an entire list of commands that can be sent to the biometric module (please refer to reference [4]). In this case however, 01 is to read an image from the sensor.

	Packet head	Device address	Packet flag	Packet length	Command	Checksum
Request	EF01	FFFFFFFF	01	0003	01	0005
Response	EF01	FFFFFFFF	07	0003	00	000A

Table 5.1: Sample ARA-ME-01 instruction

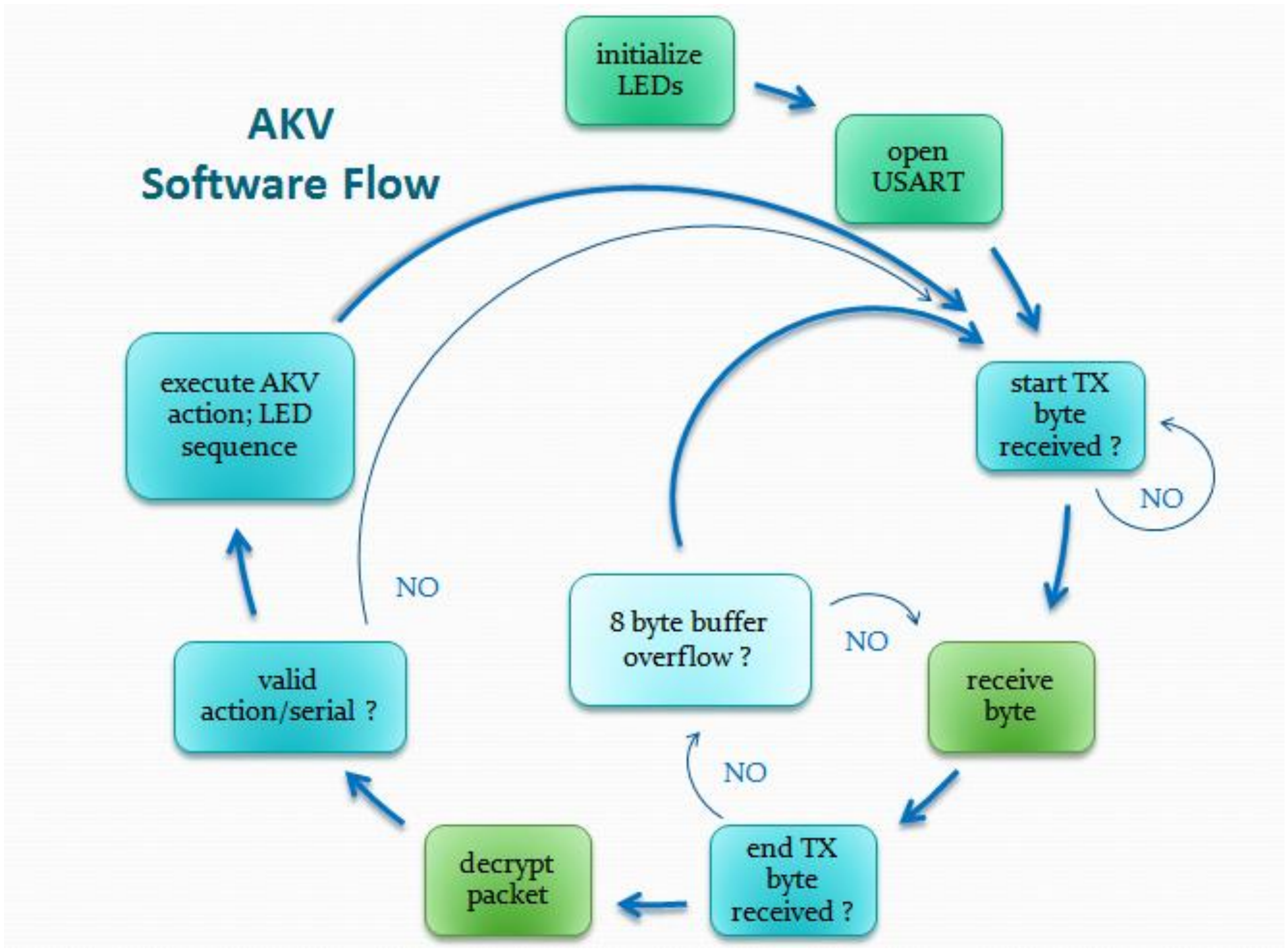


Figure 5.2: AKV Software Data Flow

Similarly to the SMART-AKK, AKV initializes the LEDs and opens the USART2 for communication. AKV begins to poll for a start transmit byte, 0xAA, from SMART-AKK. If received, it will receive the entire packet until an end transmit byte, 0xFF, is received or if a buffer overflows occurs. If all is successful, it continues to decrypt the packet and checks its validity. If it is valid, it executes the command using an LED sequence. If it is not valid, it then it goes back and polls for a start transmit byte.

The libraries, m55800_lib16 and lib_drv_16, were used in both the AKK and AKK project files.

Please Appendix IV for the source code.

6 TEST PLAN

6.1 SOFTWARE TEST PLAN

Unit tests will be performed on the AKK and AKV ARM micro controllers respectively. Testing of hardware-independent software components include:

- TEA cipher
  We will create a secure communication channel between two terminals (server and client based). The test program will prompt the user to enter four 64-bit hexadecimal numbers to be encrypted. The test program will then encrypt the numbers and show the user the encrypted versions of their input and store them to a file. The user will be able to either encrypt that file again or decrypt it. Upon decryption, the program will compare the original input to the decrypted version. If there are any variations (i.e. the comparison was not equal) the TEA cipher was incorrect and will fail the test. Otherwise, the implementation was successful and the TEA cipher has passed the test. Please see Appendix VI for TEA Cipher test program.

Testing of hardware-dependent software components include:

- Buttons and LEDs
  Tests were done to ensure button presses were being detected correctly as well as LEDs were lighting properly. A function was written to poll for button presses, store the buttons that were pressed within a specified time into an array and output the buttons that were pressed to the screen. This allows us to see if the software is correctly detecting all buttons that are pressed, if any were missed, or possibly if any buttons are not working properly. Likewise, another function was written to light LEDs. This function initially each LED on and off individually, which can be confirmed visually. This test ensures that the correct functions are being called to turn on and off the LEDs and that each LED is working.

- Biometric module
  Software components relating to the biometric module includes functions that need to send and receive commands and responses from the module.

  First, a serial port sniffer program was used to aid in understanding the communication that takes place between the test program and the biometric module. This also helped in determining the commands necessary to send to the biometric module and the associated response that would be received.

  Next, functions were written to send a command to the biometric module and to receive a response back from the module. We determined if we were sending the command correctly by comparing the response received (if any) to what would have been received if the test program was used. We then test to see if individual commands are working by executing the command through our software then connecting to the module back to the provided test software to see if the change has taken place. For example, after enrolling some fingerprints using the test software, we connect it to SMART-AKK, and send the command to delete all fingerprints. We then connect it back to the test software and see if all fingerprints have really been deleted or not.

- X-Bee wireless module
  Once the wireless modules are wired to the Atmel55 boards, each XBee can also be tested to see if they are correctly transmitting and receiving data.
  Please refer to Appendix VII for the corresponding code.

These tests also overlap into the hardware test since it will also allow us to see if hardware components are working correctly.

6.2 HARDWARE TEST PLAN

Each hardware and software component is tested independently. Afterwards, each component is interfaced one by one with the Atmel55 to ensure that the addition of each component work and do not interfere with the pre-existing system. This method will also allow us to track down any errors (if any) more easily.

In particular, the following hardware will be tested:

- Biometric module
  A demo application obtained from the manufacturer can be used to ensure that the hardware is in working condition and that it works independently before it is interfaced with the Atmel55.

- X-Bee wireless modules
  The software included in the X-Bee development kit provides a range test. Clearly, this test will allow us to see the range of communication between the X-Bees but it will also allow us to test if each X-Bee is in working condition.

The functional tests in Table 6.1 describe the performed tests in more detail, which also allows us to ensure that SMART-AKK is working correctly:

Test	Action	Expected Result
Biometric Module		
Invalid fingerprint swipe	Swipe finger that has not yet been enrolled	Biometric module should not find a match
Valid fingerprint swipe	Swipe finger that has been enrolled	Biometric module should return the id of the fingerprint (id it was stored as)
Enroll fingerprint	Send command to enroll fingerprint	Fingerprint is enrolled (verify with test application)
Delete fingerprints	Send command to delete fingerprints	All fingerprints are deleted from memory (verify with test application)
ARM Board		
Polling for button press	Press two buttons consecutively	Two buttons are detected
Stick buttons	Press and slightly hold one button	Program only detected as one button press
Valid sequence	Press a valid sequence of buttons (IE. Double press button 1)	Sequence detected as valid
Invalid sequence	Press an invalid sequence of buttons (IE. Double press button 4)	Sequence detected as invalid
LED verification	Perform a number of steps on SMART-AKK (swipe finger, press button etc).	LEDs light accordingly as each step is executed successfully (refer to Table 2.1)
TEA		
Encryption and decryption	SMART-AKK sends an encrypted packet to AKV and AKV decrypts the packet	The decrypted packet on AKV's side should be the same as the packet before encryption on SMART-AKK's side.
Wireless		
Range test	Use the development kit to test the range of the X-Bees	A range of approximately 10m should be allowed
Working condition	Power using the development kit to determine if X-Bees are in working condition	Light on the development kit should be blinking

Table 6.1 functional tests performed

7 RESULTS OF EXPERIMENTS

In terms of the functional tests that were described above, all tests produced expected results.

Additional results are as follows:

- Biometric module
    - Commands sent to the biometric module needed to be sent byte by byte so that there was no confusion detected by the biometric module with the packet format.
    - To enroll a fingerprint, it is necessary to continuously send the ‘get image from buffer’ command to the module until it detects a finger swipe. Once the command is sent, the module will immediately carry out the command; there cannot be a delay between the sent command and finger swipe. Therefore, it is necessary to continuously send the command until the module accurately detects the image from the sensor.
    - When swiping your fingerprint, it is necessary to swipe in the same direction and format as when it was enrolled. Otherwise, even if an enrolled finger is being swiped, the module will not detect a match.
    - From our own trial runs:
        1. 9/10 times, a valid fingerprint was detected as invalid.
        2. 10/10 times, an invalid fingerprint will be detected as invalid.
    - Sometimes when swiping your finger, the biometric module will get ‘stuck’ (the module would stop responding to any requests). The actual cause of this problem has not been fully determined; however, we believe it to be associated with the way the finger is swiped on the sensor (IE. The finger was swiped at an angle).

- Push buttons

  We had hoped to get button presses using internal interrupts on the Atmel55, but due to problems with setting up the interrupt service routine and after consulting with the lab instructor, we resorted to polling. In reality however, we would want to use interrupt driven button presses to have a more efficient product.

  With polling, we initially had a problem with sticky button presses; a single button press was sometimes detected as two button presses because the USART status did not change fast enough or because the button was held down too long (while the software was checking for a second button press). As a result, we implemented a short delay factor after each button was pressed just to account for these ‘sticky’ buttons (after one button was pressed, it would wait a little while before looking for the next button press). Because this delay is fairly short, it did not affect the accuracy in obtaining button presses, it only aided in removing the problem of ‘sticky’ buttons. Having LED indicators when a button was pressed helped in determining if a button press was detected or not.

- X-Bee
  Wireless communication has been successfully achieved. One of the first experimentations done with the X-Bees included the range test using the software included in the development kit. We were able to have a distance of approximately 10m between the two X-Bees before the program occasionally complained of lost packets. Once the X-Bees were configured and integrated onto SMART-AKK and AKV, the functions in Appendix VII were written to help test the X-Bee functionality. These functions allowed us to test if there were any problems in transmitting and receiving data and/or if the functions were correctly written. No problems were detected; data that was sent and received were identical. Additionally, testing also indicated that the X-Bee is able to receive up to 256 in hexadecimal.

- TEA
  The TEA functions use a Feistel Cipher which uses operations from the following orthogonal algebraic groups – XOR, ADD and SHIFT. It encrypts 64-data bits at a time using a 128-bit key. The TEA functions performed exactly as planned. The algorithm achieves complete diffusion because through one change in plain text it generates 32-bit differences in cipher text in six rounds. Please refer to Appendix VII.

8 REFERENCES

[1] Shephard, Simon J. The Tiny Encryption Algorithm. Taylor & Francis.

[2] Minderman, Nancy. CMPE 490 Resources. [Online] Available: http://www.ece.ualberta.ca/~CMPE490/winter2010/resources.html Accessed 2010.

[3] Aratek Biometrics Technology Co. Ltd. Fingerprint module specification. [Online] Available: http://www.sparkfun.com/datasheets/Sensors/Biometric/ARA-ME-2510.pdf Accessed January 2010.

[4] Aratek Biometrics Technology Co. Ltd. Fingerprint module specification. [Online] Available: http://www.sparkfun.com/datasheets/Sensors/Biometric/fingerprint%20module-2510.doc Accessed January 2010.

[5] Digi.com. (2009). XBee & XBee-PRO ZB. [Online] Available: http://www.digi.com/pdf/ds_XBeezbmodules.pdf Accessed January 2010.

[6] MaxStream. (2006, October 13). XBee / XBee-PRO OEM RF Modules. [Online]. Available: http://www.libelium.com/squidbee/upload/3/31/Data-sheet-max-stream.pdf Accessed January 2010.

[7] Sparkfun Forum. (2008) Fingerprint Slide Scanner. [Online]. Available: http://forum.sparkfun.com/viewtopic.php?f=14&t=12432&start=60 Accessed March-April 2010.

[8] Minderman, Nancy. CMPE490 Project – FLASH Tutorial. [Online] Available: http://www.ece.ualberta.ca/~cmpe490/winter2009/project/flash.html Accessed April 8, 2010.

Quick start manual

Necessary parts:

i. 2 Power supplies
ii. 2 ARM boards & power cables
iii. 5 Power cables (3 red, 2 black)
iv. Smart-AKK prototype board (serial and header cable attached)
v. AKV prototype board (header cable attached)

Setup:

Flash Projects

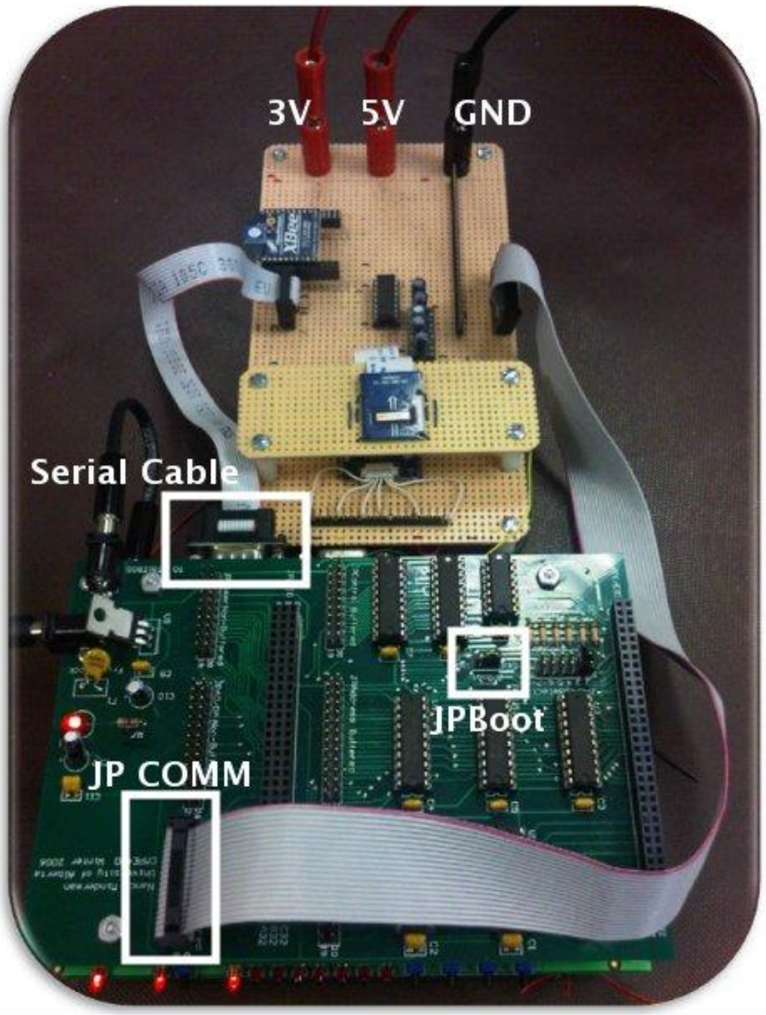
- Follow the flash instructions indicated in the FLASH tutorial (Reference section [8]).
- The SMART-AKK ARM board should be flashed with project SMART, and the AKV board with project VEHICLE.

Assembling Projects (After projects have been flashed)

SMART-AKK (AKK)

1. Connect the serial cable of AKK to serial port A of the ARM board
2. Connect the header cable of AKK to the JP COMM buffer on the ARM board. Ensure that the arrow on the cable connects to pin 1 of the board.
3. Connect power cables of AKK. From left to right, the board should be powered as follows: 3V, 5V, GROUND.
4. Ensure that the JPBoot jumper is switched to USR.
5. Power the ARM board. You should see all (with the exception of one) of the LEDs flash simultaneously.
6. Turn on the power supply for the AKK.

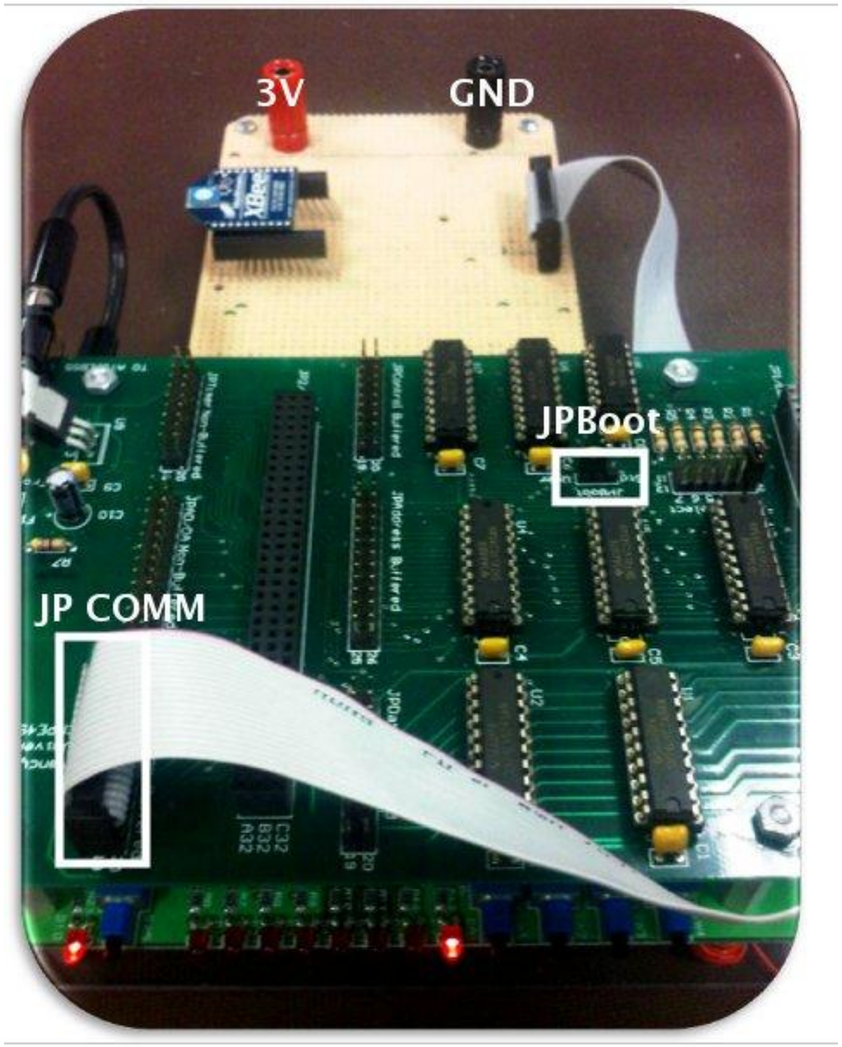
Your set up should look similar to the following:



AKV

1. Connect the header cable of AKV to the JP COMM buffer on the ARM board. Ensure that the arrow on the cable connects to pin 1 of the board.
1. Connect power cables of AKV. From left to right, the board should be powered as follows: 3V, GROUND.
2. Ensure that the JPBoot jumper is switched to USR.
3. Power the ARM board. You should see all (with the exception of one) of the LEDs flash simultaneously.
4. Turn on the power supply for the AKV.

Your set up should look similar to the following:



Demonstration

1. Begin by powering both the SMART-AKK and AKV (This should have already been done if you followed the 'Setup' instructions).
2. Review LED indicators in Table 2.1
3. Ensure that SMART-AKK is ready; LED8 should be on.
4. Ensure that the biometric module is ready; the light on the biometric module is blinking.
5. Swipe an enrolled finger on AKK.
6. Press a valid command sequence.
7. If you pressed an action command, watch the associate LED sequence play on AKV.
8. If you pressed the enrolment sequence, swipe the new fingerprint when the biometric light starts blinking. Swipe the same finger a second time when the light starts blinking again.
9. If you pressed the command to delete all fingerprints, continue to enroll the new master.

i BACKUP PLANS

There exist two major parts of the design with the possibility of failing: biometric authentication and wireless communication.

If there is a biometric failure, (i.e. module is unable to communicate with the SMART-AKKs Atmel55) the security feature of the SMART-AKK will still be maintained. The user can be authenticated by entering a pre-determined password on SMART-AKK. If the password passes, the AKK will proceed as previously described to generate a code and prepare the packet for wireless transmission. AKV would not be affected by the change in authentication method.

If wireless communication fails, the user can us a manual key to control the vehicle.

ii OPTIONS / EXTENSIONS

To make the interaction between the SMART-AKK and the vehicle simulator more realistic, additional features to the project could include sound functionality, through horns or beeps to confirm AKK button presses. The AKV can be made to work (i.e. Able to turn the ignition on/off, drivable vehicle) between a certain time period, when this time period expires the AKV goes into a sleep mode. AKV can be awoken from sleep mode when it receives a predetermined sequence of button presses from SMART-AKK.

The SMART-AKK is not limited to communication with an AKV. The SMART-AKK can be applied to any situation requiring greater access security. For example, it can be used to secure a wireless channel in an office building's security system or to provide additional security when opening a safe.

Since SMART-AKK and AKV are prototypes, the size of the development boards and modules are extremely large to be convenient. Therefore, a more exciting extension would be to modify SMART-AKK so that it would be the size of a regular SmartKey.

CIRCUIT DIAGRAM OF HARDWARE

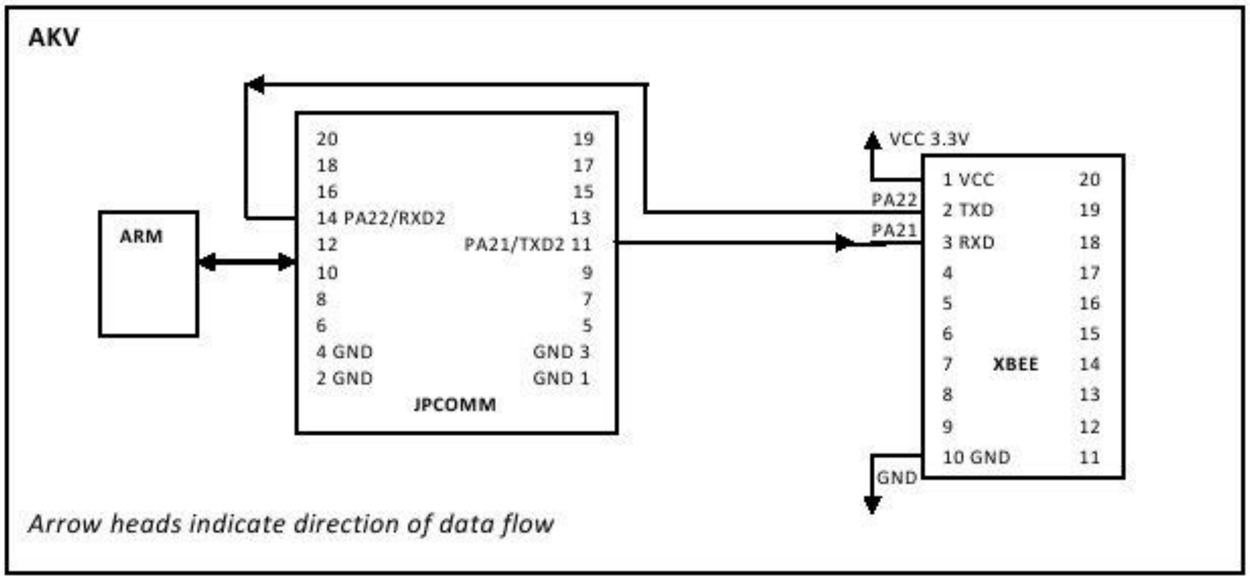


Figure III.1 Circuit diagram of AKV

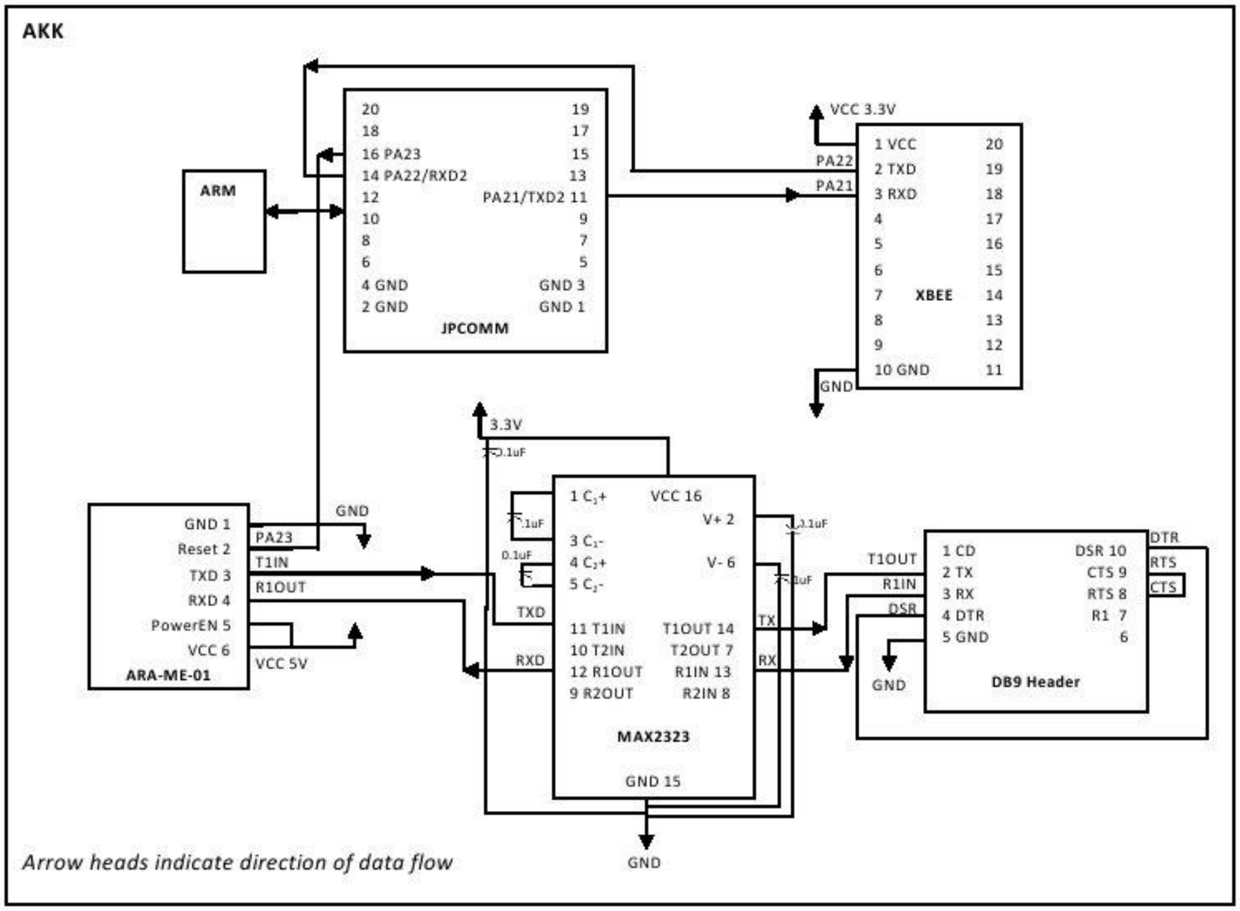
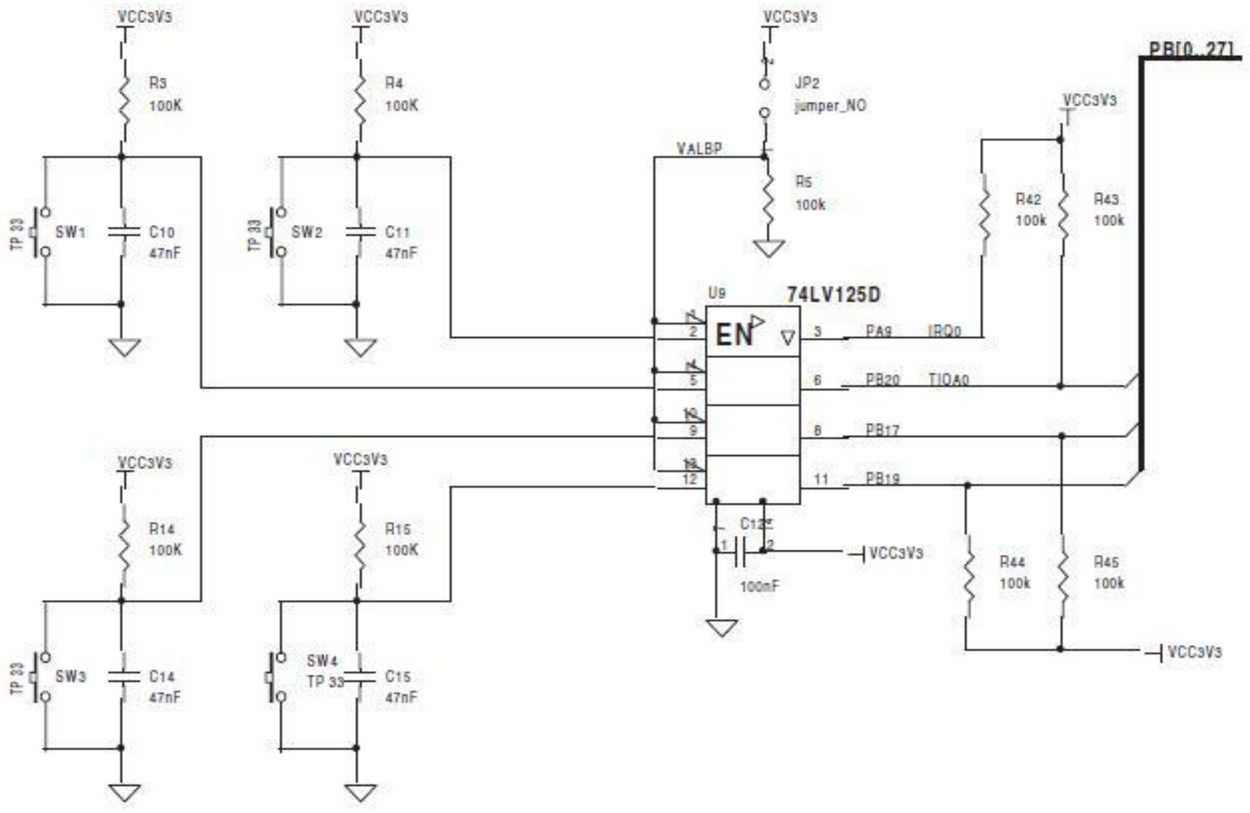
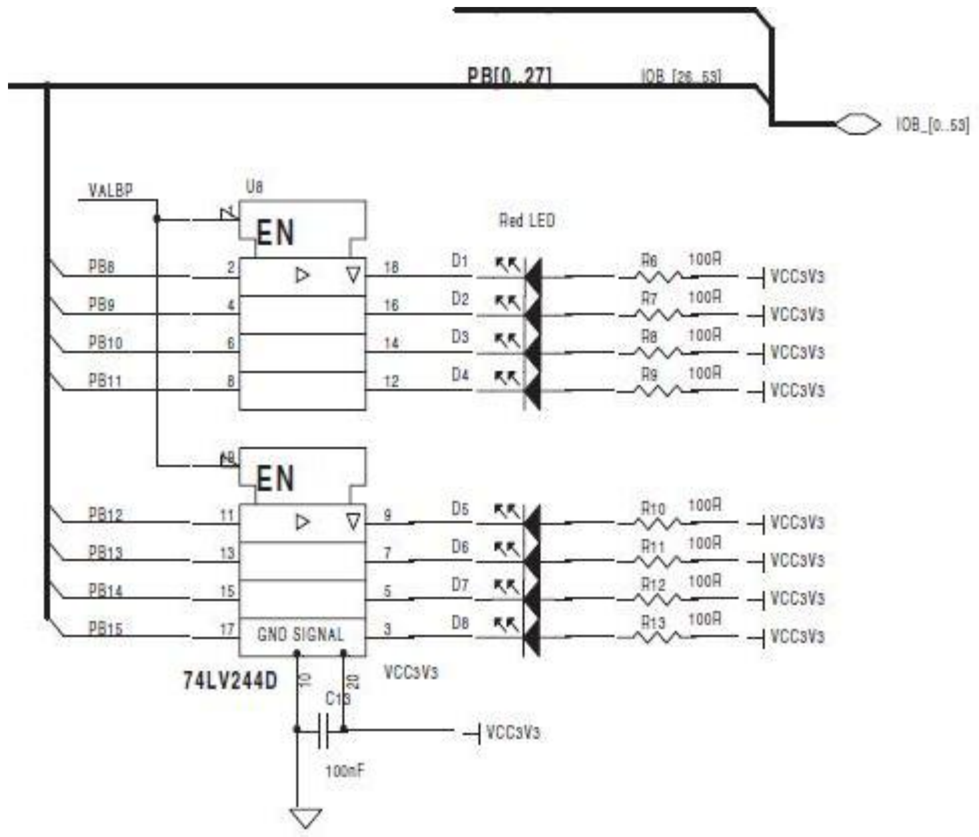


Figure III.2 Circuit diagram of SMART-AKK

Push Buttons configuration on Atmel55

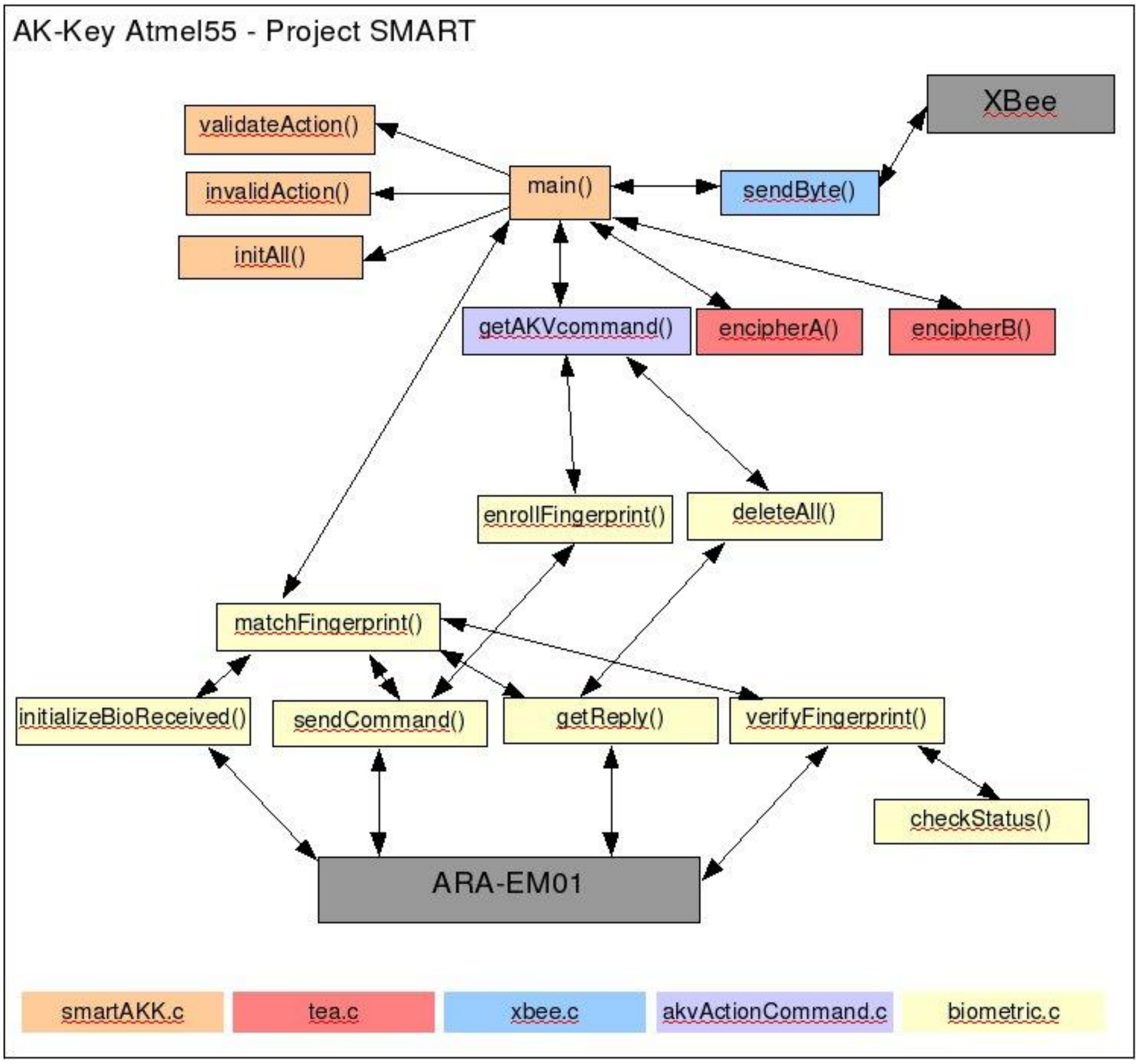


The 8 LEDs connect to PB [8 .. 15] on ATMEL55



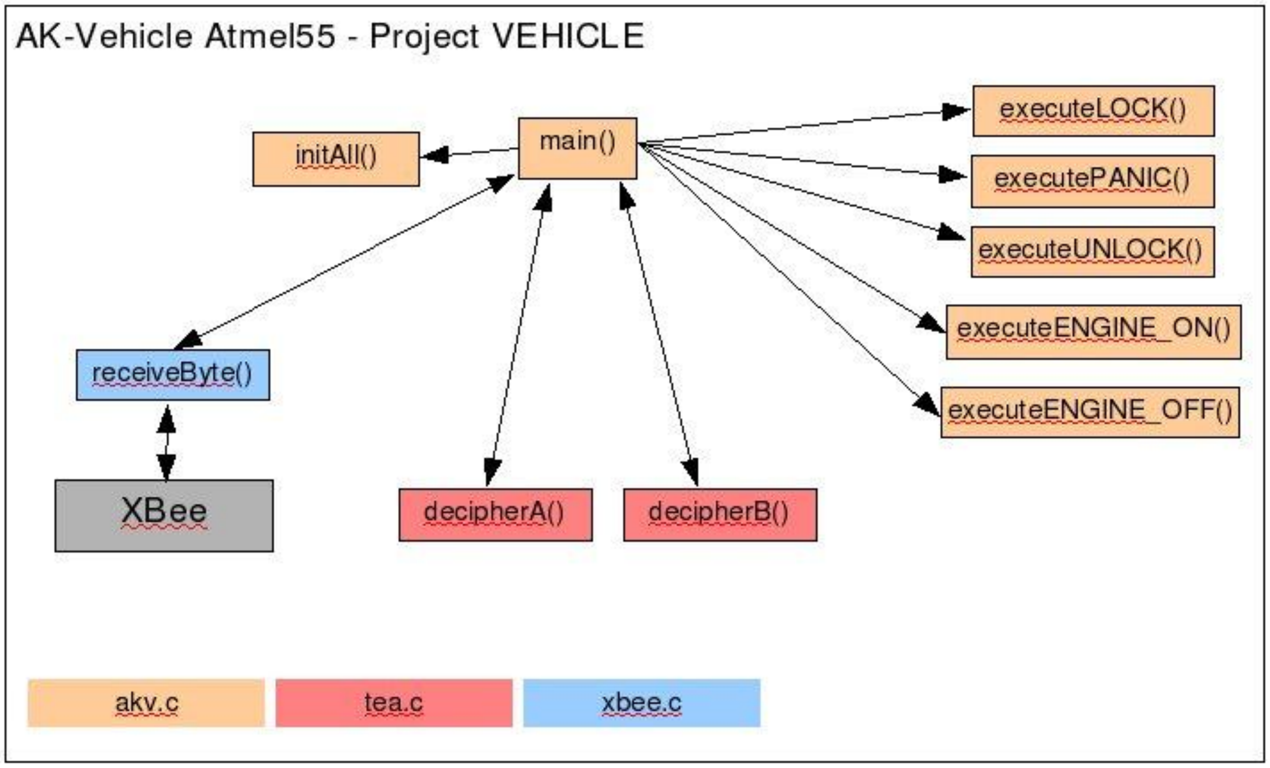
Source code is attached in a .zip file.

BLOCK DIAGRAM OF INTERACTIONS



SMART - AKK
(Status for all: Tested and passed)

File Name	**Description**
smartAKK.c	Main program of SMART-AKK
smartAKK.h	Header for main function of SMART-AKK
XBee.c	Contains the necessary functions to communicate with the XBee module and send data to AKV
tea.c	Encrypts data using TEA
biometric.c	Contains the necessary functions to communicate with the ARA-ME-01 module
biometric.h	Header file for biometric module
akvActionCommand.c	Valid AKV action command numbers and serial number information



AKV
(Status for all: Tested and passed)

File Name	**Description**
akv.c	Main program of AKV
XBee.c	Contains the necessary functions to communicate with the XBee module and receive data to AKV
tea.c	Decrypts data using TEA

APPENDIX V XBee CONFIGURATION APPLICATION NOTE

Please refer to the application notes posted here:
http://www.ece.ualberta.ca/~elliott/cmpe490/appnotes/2010w/XBee_Wireless_Communication/

For a detailed instruction on how the XBees were configured.

```
/* teaTEST.c */
int main( void )
{
    unsigned int aPacket = 0x11111111;
    unsigned int addPacket = 0x4;

    unsigned int VIN0 = 0xAA;
    unsigned int VIN1 = 0x11;
    unsigned int VIN2 = 0xBB;
    unsigned int VIN3 = 0x22;

    unsigned int serialNum = 0x12345678;
    unsigned int serialCount = 0x8;


    unsigned int buttonV0 = aPacket;
    unsigned int buttonV1 = serialNum;

    unsigned int encV0 = 0x0;
    unsigned int encV1 = 0x0;

    unsigned int decV0 = 0x0;
    unsigned int decV1 = 0x0;


       while(1)
       {
              buttonV0 = buttonV0 + addPacket;
              encV0 = buttonV0;
              buttonV1 = buttonV1 + serialCount;
              encV1 = buttonV1;

              printf("BEFORE ENCRYPTION\t\t[%X %X]\t", buttonV0, buttonV1);

              encV0 = encipherA(buttonV0, buttonV1, VIN0, VIN1, VIN2, VIN3);
              encV1 = encipherB(buttonV0, buttonV1, VIN0, VIN1, VIN2, VIN3);

              if((encV0 == buttonV0) || (encV1 == buttonV1))
              {
                     printf( "\t\tTEA ENCRYPTION TEST FAILED\n" );
                     break;
              }
```

```
            else
                   printf( “\t\tTEA ENCRYPTION PASSED\n” );


            printf(″TEA ENCRYPTED\t[%X %X]\n″, encV0, encV1);

            decV0 = decipherA(encV0, encV1, VIN0, VIN1, VIN2, VIN3);
            decV1 = decipherB(encV0, encV1, VIN0, VIN1, VIN2, VIN3);


            if((decV0 != buttonV0) || (decV1 != buttonV1))
            {
                   printf( “\t\tTEA DECRYPTOPN TEST FAILED\n” );
                   break;
            }
            else
                   printf( “\t\tTEA DECRYPTION PASSED\n” );


            printf(″AFTER ENCRYPTION\t[%X %X]\n″, encV0, encV1);
      }
   return(0);
}
```

```
/* akkTESTXBee.c */

#include <string.h>
#include <time.h>

#include ″parts/m55800/eb55.h″
#include ″parts/m55800/lib_m55800.h″
#include ″periph/stdc/std_c.h″
#include ″periph/usart/usart.h″

# define BAUDS38400   (32000000 / (16 * 38400))

int main( void )
{
    unsigned int command = 0x5;

    at91_usart_open(&USART2_DESC, (u_int) US_ASYNC_MODE, (u_int) BAUDS38400, 0);

    while(1)
    {
       //Check uart status and get transmit status to see if its ready
       if (at91_usart_get_status(&USART2_DESC) & US_TXRDY)
       {
           if(at91_usart_write(&USART2_DESC, command))
              printf( “SMART-AKK XBee PASSED TEST\n” );
           else
              printf( “SMART-AKK XBee FAILED TEST\n” );
       }
    }
    return(0);
}
```

```
/* akvTESTXBee.c */

#include <string.h>
#include "periph/stdc/std_c.h"

#include "parts/m55800/lib_m55800.h"
#include "drivers/capture/capture.h"
#include "drivers/wait/wait.h"
#include "parts/m55800/eb55.h"
#include  "periph/usart/usart.h"

#define BAUDS38400       (32000000 / (16 * 38400))    //* CD = 52

int i;

int main( void )
{
    unsigned int command = 0x5;
    unsigned int aByte = 0x0;

    at91_usart_open(&USART2_DESC, (u_int) US_ASYNC_MODE, (u_int) BAUDS38400, 0);

    while(1)
    {
       aByte = retrieveData();

       if(aByte != 0x0)
       {
           if(aByte != command)
           {
              printf("AKV XBee FAILED TEST\n”);
              break;
           }
           else
              printf( “AKV XBee PASSED TEST\n”);
       }
    }
    return(0);
}
```