ATM Banking System

Objective :-

Apply and use all we learned about AVR (internal peripherals and communication protocols) and apply them in one useful project (payment system)

Hardware component Level :

In this project what we will need

1-2 MCUs (one of them will represent the ATM node and the other one will represent the Card node

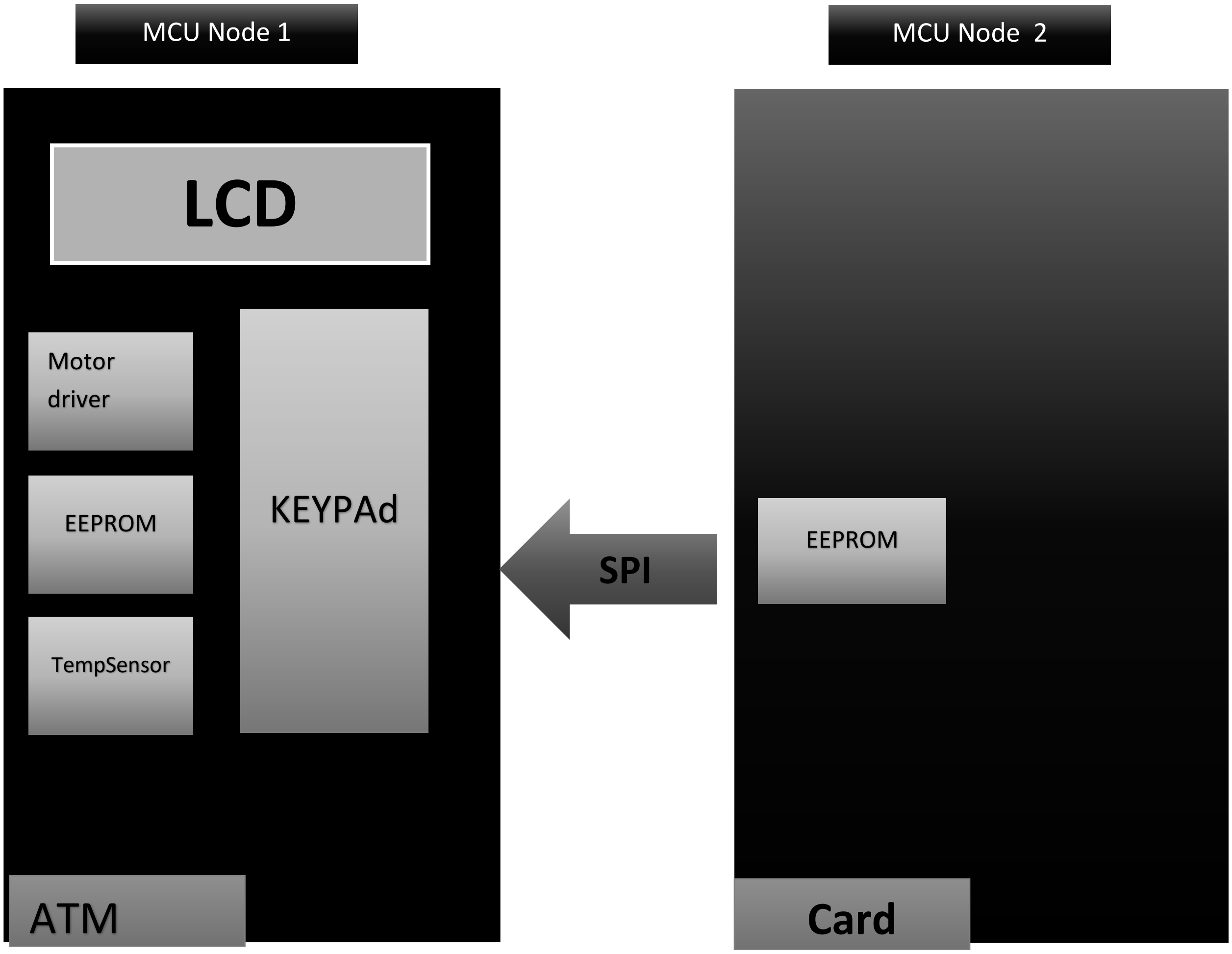
2-Two External EEPROM (one for ATM act as server data base and one for card hold card Data like PAN , Name , CVV)

3-Motor And H-bridge (act as cache out).

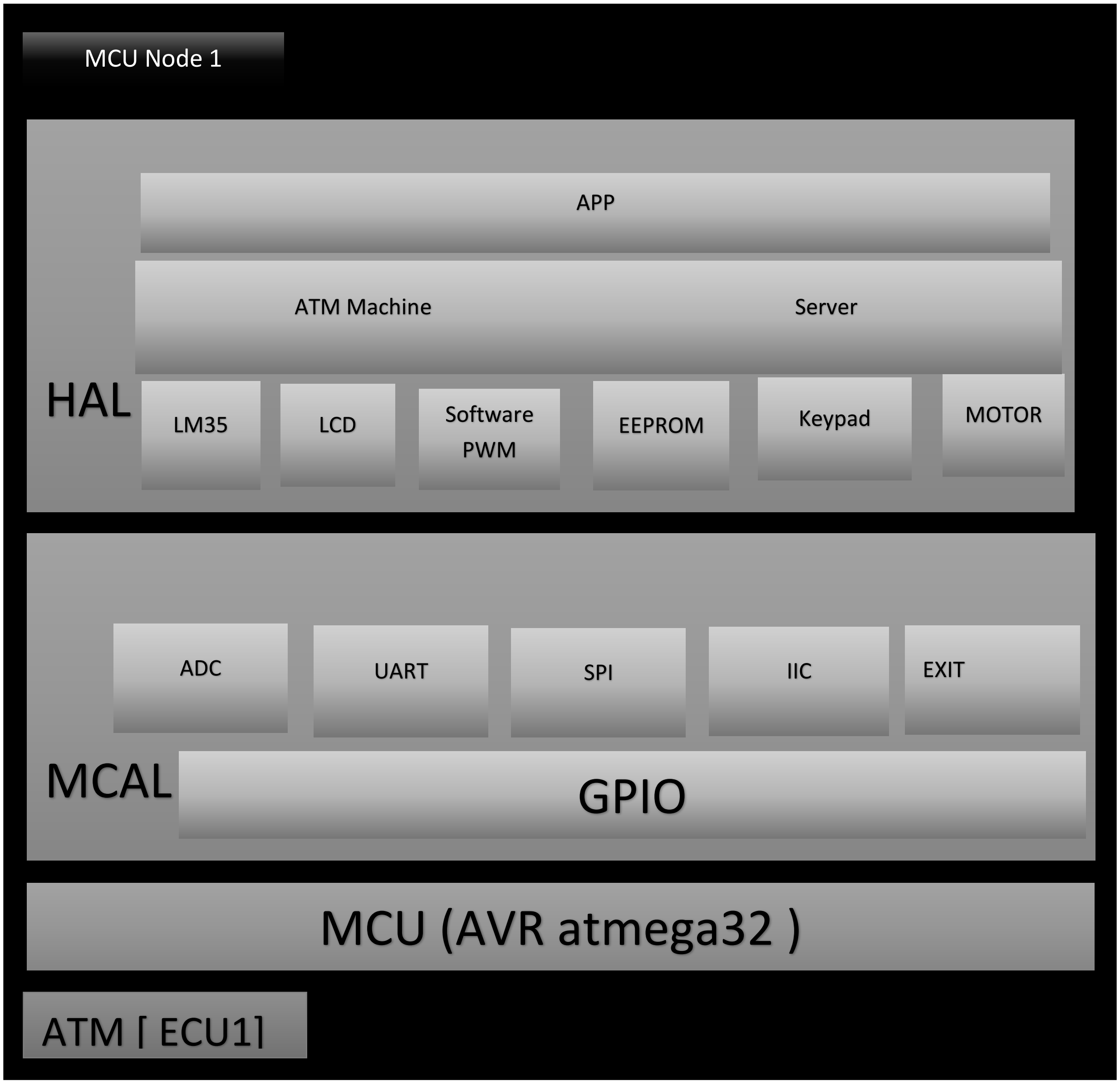
4-KEYPAD for ATM machine and LCD AS a user interface

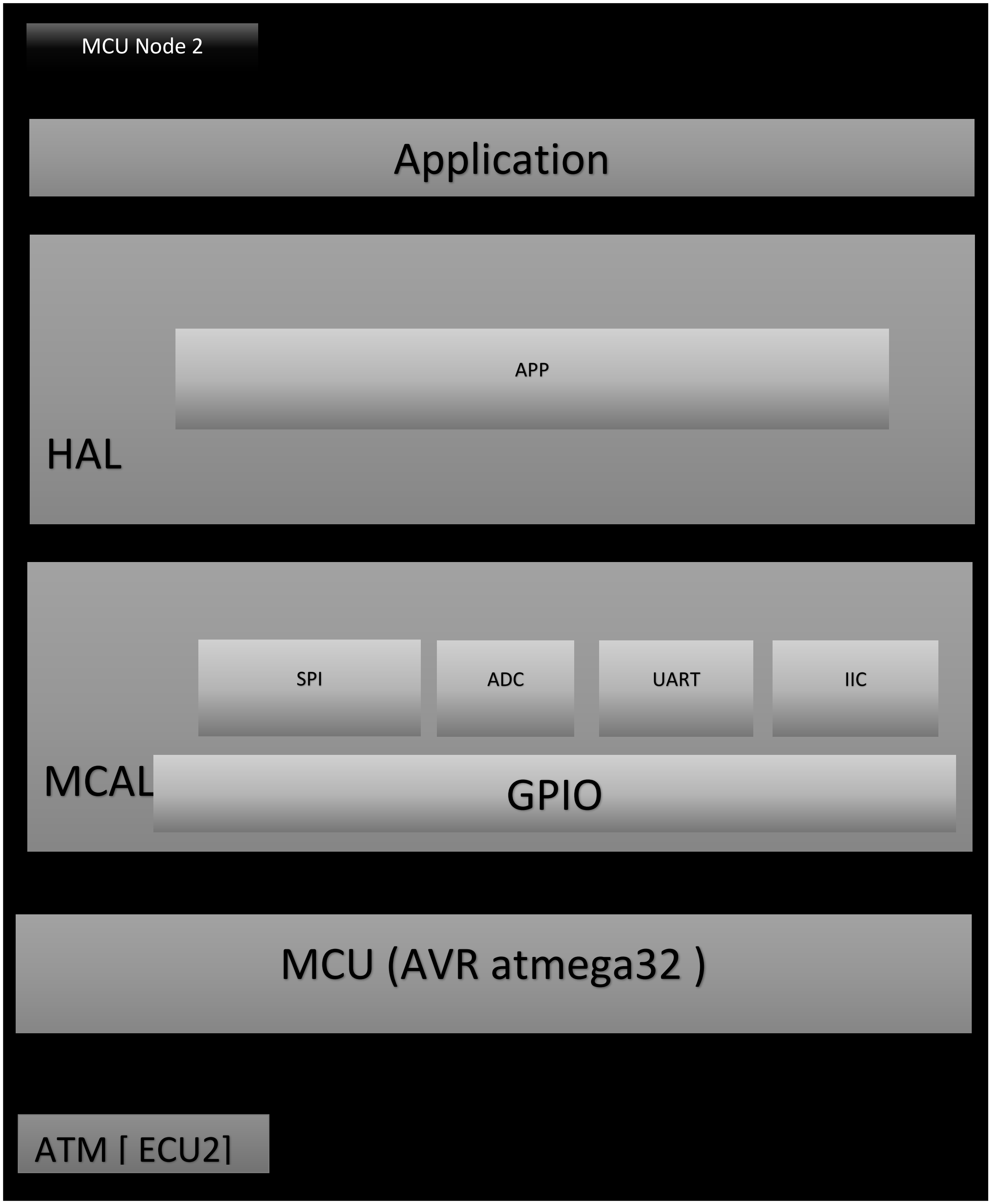
5-Temperature sensor act as hardware guard against hacking

Hardware Component Layer



Now after we had a closer look for the hardware component level so now we're ready to go to the next level of static design (software component level)




MCU Node 2
Application
APP
HAL
SPI
ADC
UART
IIC
MCAL
GPIO
MCU (AVR atmega32)
ATM [ECU2]

Release 1 :

For ATM Machine Feature :

1- Two Modes of operation USER OR ADMIN mode
2- Add Or Remove any new card
3- Search for any installed card in database with Card holder name or card number
4- Admin mode can handle all Machine security and specific parameters [Max Temp – Admin user name or admin password]
5- Add another specific Commands that will be incremented in another released

For Card Machine Features :

1- Working in two modes [admin / user] mode
2- Ability to change any of Card parameter With admin mode

Brief Intro about this ATM Banking system

First of all I want to discuss some of important information about any ATM machine

Today all of us may be or not work with ATM in our normal daily live and of course it’s work as a secure and separated pocket the help us any where and every where

So now we need how this device work to convert it into software that can act as real one of them

First we need to separated devices the first one is the Card that we hold it and the ATM machine that connected to the Bank server that hold all information about our account into a secured database

Of course we now figured out that we need two MCUs one of them act as Card and the another one act as ATM machine

But in real world we can know where is the complex part

Card or ATM of course if we have a closer look for each one of them we can see that ATM machine is bigger in size and Data that can be deal with every day form inserting card till your money withdraw if your balance is enough

So with out any further ado let’s cut to the chase

Here we have a programming mode in each one of them so we can at run time handle any configurations

Only the case that 2 MUUs that can communicate with each other if 2 MCUs are in Operating mode

- We simulate card inserted with Button
- We simulate ATM Hardware security by Temp sensor
- We simulate success transaction operation with Motor
- We have Two ways to act with ATM machine FOR user can use Keypad and LCD for ADMIN will use Terminal
- All data that printed out into the LCD is also printed out into Terminal for Admin
- We simulate EEPROM as a server Database that hold all user information
- We simulate Card Reader as SPI protocol

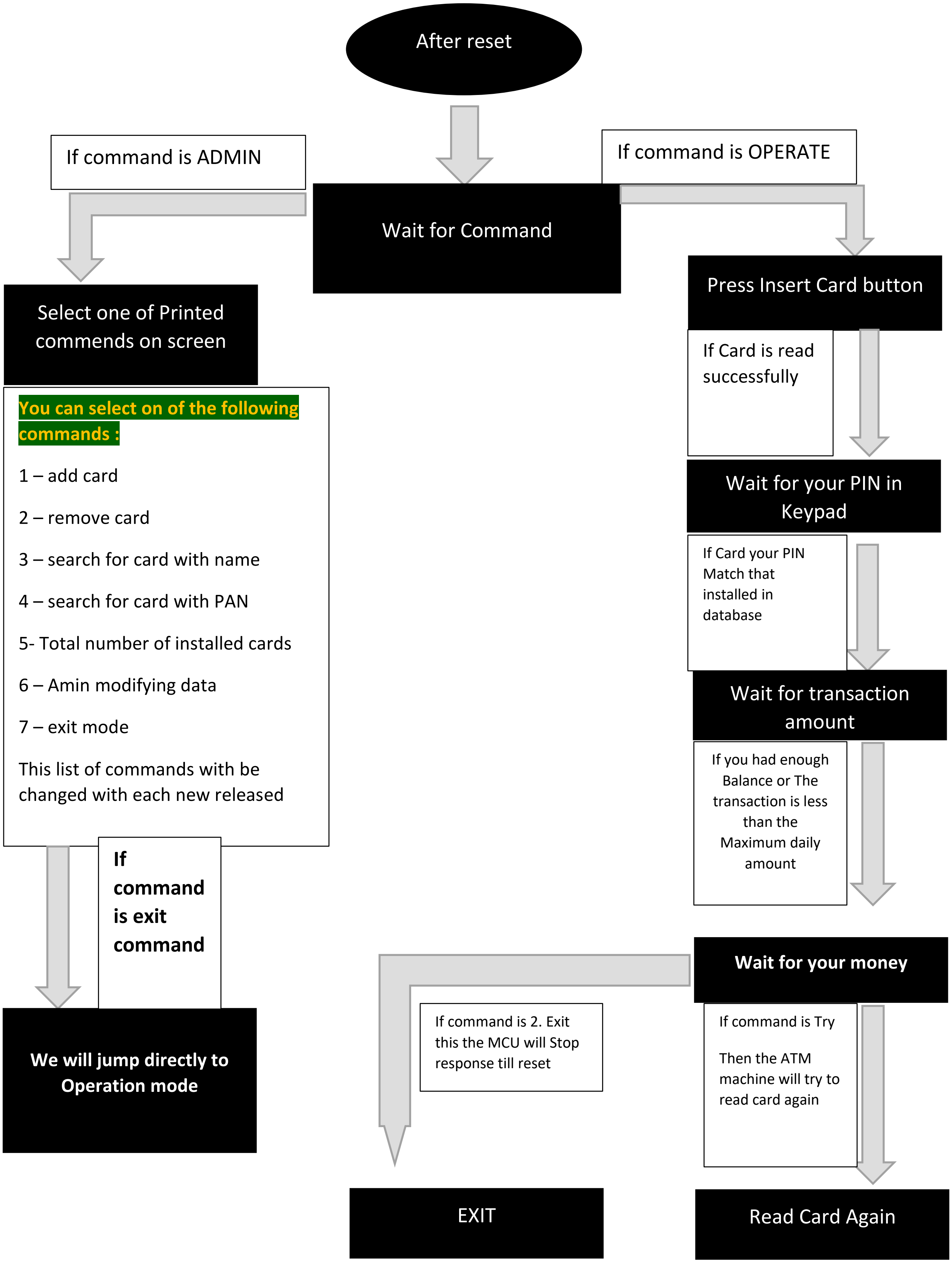
I think now we have got the main idea about need of each Element in our system

Now every thing is ready to explain Software

We have to options to specify the operation for ATM and Card (State machine / flow chart)

Se I prefer to use flow chat so we can explain everything in details if we need

Let’s get started with ATM machine operation



After reset

Wait for Command

If command is ADMIN

Select one of Printed commends on screen

You can select on of the following commands :

1 – add card

2 – remove card

3 – search for card with name

4 – search for card with PAN

5- Total number of installed cards

6 – Amin modifying data

7 – exit mode

This list of commands with be changed with each new released

If command is exit command

We will jump directly to Operation mode

If command is OPERATE

Press Insert Card button

If Card is read successfully

Wait for your PIN in Keypad

If Card your PIN Match that installed in database

Wait for transaction amount

If you had enough Balance or The transaction is less than the Maximum daily amount

Wait for your money

If command is 2. Exit this the MCU will Stop response till reset

EXIT

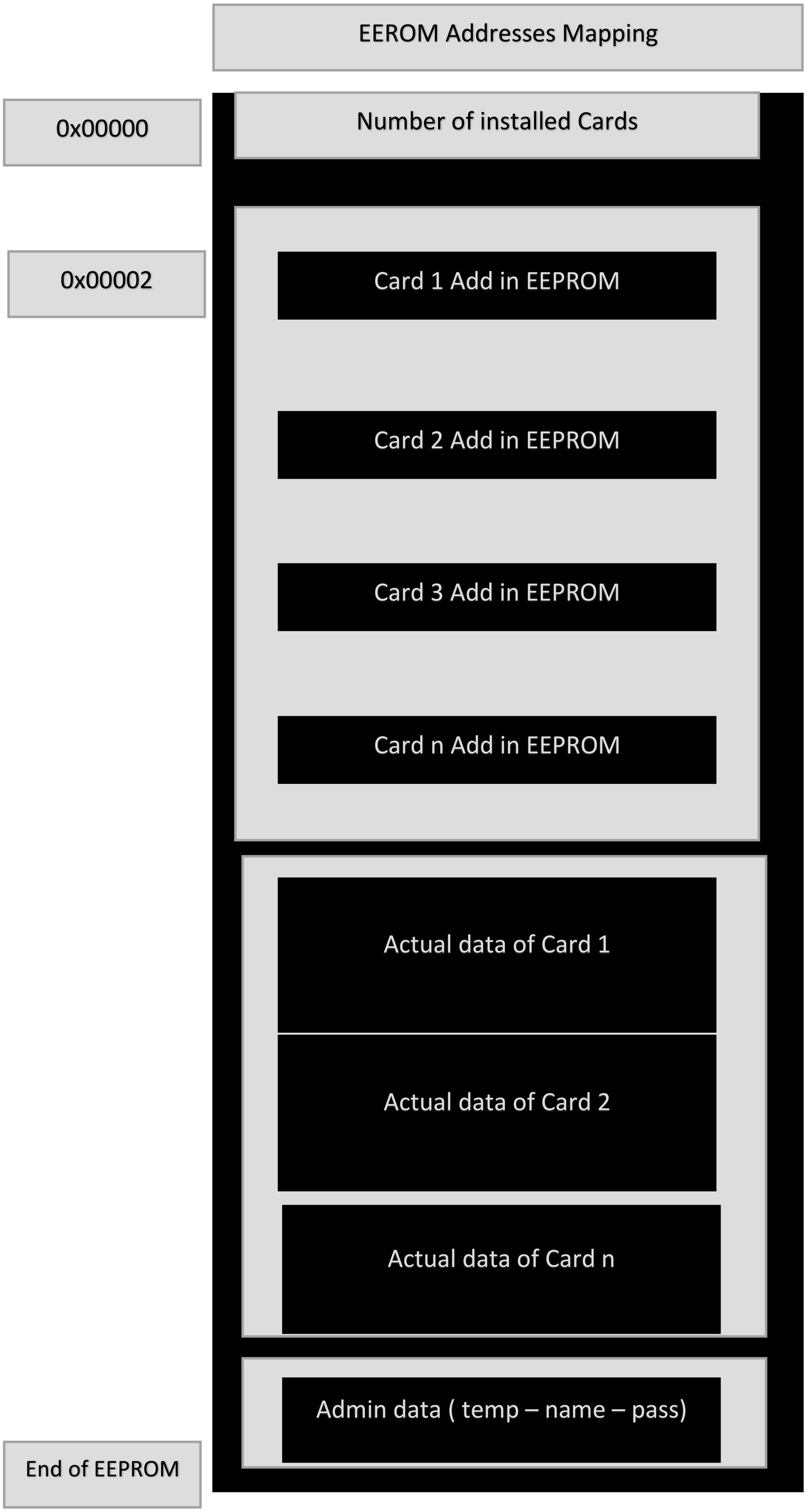
If command is Try

Then the ATM machine will try to read card again

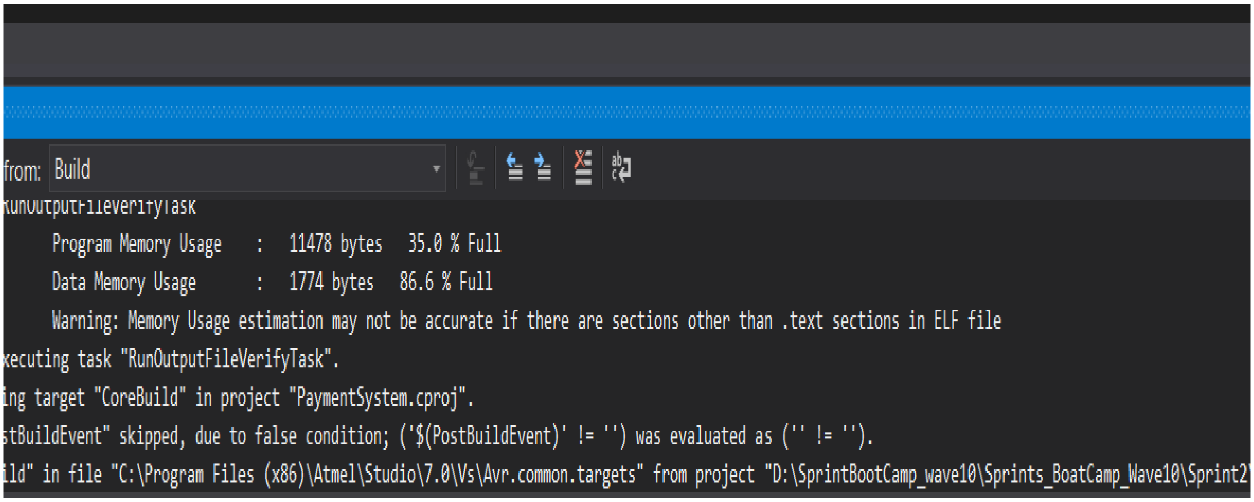
Read Card Again

Issues that we faced during build this project

Problem	Solved	Unsolved
Hardware requirement that H=Bridge EN motor not mapped into any existing PWM in the AVR MCU	**Here we can use two methods:** **1-** Use PWM with GPIO normal mode option but inside ISR we can toggle any other GPIO Pin **2-** We can use CTC mode which is more stabile that the first solution (so I prefer this solution) But you need to know that any solution will add interrupt overhead into your Project	
Enormous String (for user interface)that use all ram Space of course this type of literal string use all Ram because AVR didn't define .ro section in Linker	**AVR proved** #include <avr/pgmspace.h> library that help us to put this string into Flash (this string is not changed through the program execution But this will reduce the efficiency of the MCU due to the fact of Flash is very slow than Ram <AT> <the end> < I'll proved to you some of Photos that will show you the difference before and after use this solution	
ADC need to run Periodically (temperature sensor) due to security check operation	Here the solution for this problem is very simple we can use timer1 as a trigger for ADC to start its operation and inside ISR of ADC we can check	
EEPROM Handling	Of course dealing with big data using EEPROM is big performance issue so this sections is under development to get the best way for searching and sorting inside this EEPROM	Under development


EEROM Addresses Mapping
0x00000
Number of installed Cards
0x00002
Card 1 Add in EEPROM
Card 2 Add in EEPROM
Card 3 Add in EEPROM
Card n Add in EEPROM
Actual data of Card 1
Actual data of Card 2
Actual data of Card n
Admin data (temp – name – pass)
End of EEPROM

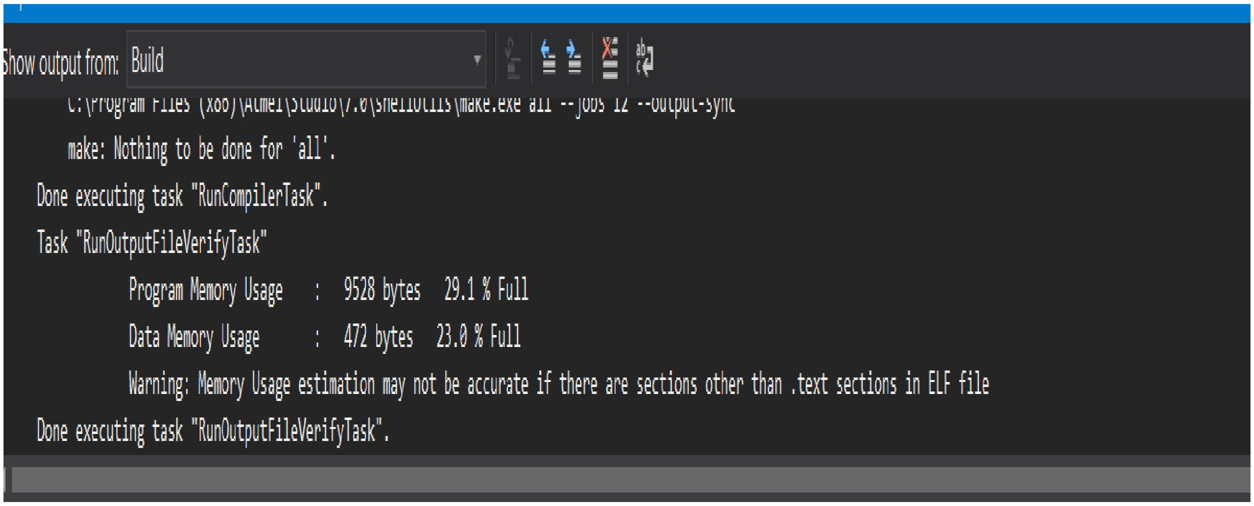
Before Store literal string into flash



from: Build

```
RunOutputFileVerifyTask
     Program Memory Usage    :   11478 bytes   35.0 % Full
     Data Memory Usage       :   1774 bytes   86.6 % Full
     Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
xecuting task "RunOutputFileVerifyTask".
ing target "CoreBuild" in project "PaymentSystem.cproj".
tBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
ild" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "D:\SprintBootCamp_wave10\Sprints_BoatCamp_Wave10\Sprint2
```

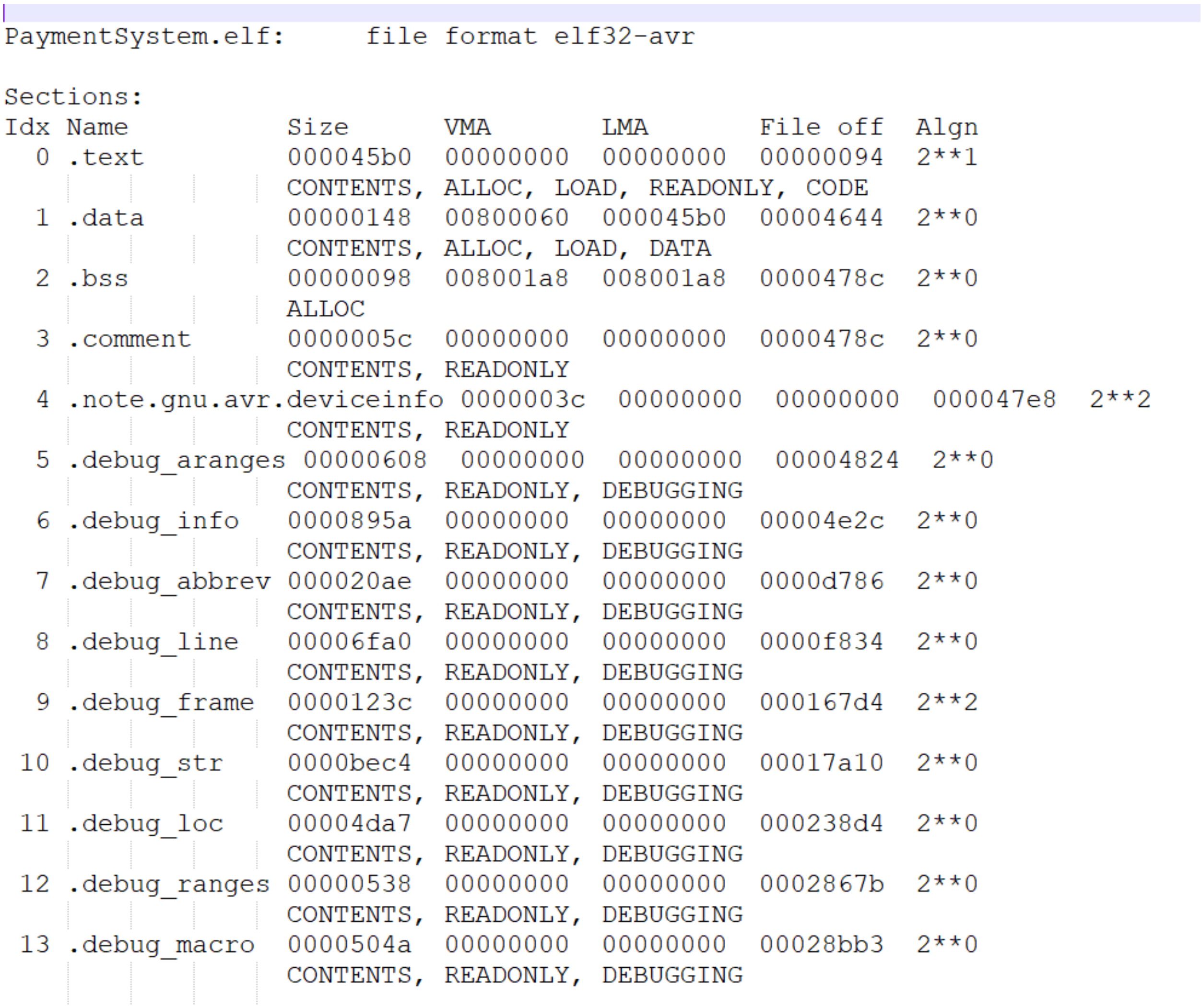
After Store the same number of bytes into Flash memory with performance penalty



Show output from: Build

```
      C:\Program Files (x86)\Atmel\Studio\7.0\shellUtils\make.exe all --jobs 12 --output-sync
      make: Nothing to be done for 'all'.
   Done executing task "RunCompilerTask".
   Task "RunOutputFileVerifyTask"
            Program Memory Usage    :   9528 bytes   29.1 % Full
            Data Memory Usage       :   472 bytes   23.0 % Full
            Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
   Done executing task "RunOutputFileVerifyTask".
```

As we see here defined sections inside AVR don’t mention for .ro data for Literal strings



```
PaymentSystem.elf:     file format elf32-avr

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000045b0  00000000  00000000  00000094  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000148  00800060  000045b0  00004644  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000098  008001a8  008001a8  0000478c  2**0
                  ALLOC
  3 .comment      0000005c  00000000  00000000  0000478c  2**0
                  CONTENTS, READONLY
  4 .note.gnu.avr.deviceinfo 0000003c  00000000  00000000  000047e8  2**2
                  CONTENTS, READONLY
  5 .debug_aranges 00000608  00000000  00000000  00004824  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_info   0000895a  00000000  00000000  00004e2c  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_abbrev 000020ae  00000000  00000000  0000d786  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_line   00006fa0  00000000  00000000  0000f834  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_frame  0000123c  00000000  00000000  000167d4  2**2
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_str    0000bec4  00000000  00000000  00017a10  2**0
                  CONTENTS, READONLY, DEBUGGING
 11 .debug_loc    00004da7  00000000  00000000  000238d4  2**0
                  CONTENTS, READONLY, DEBUGGING
 12 .debug_ranges 00000538  00000000  00000000  0002867b  2**0
                  CONTENTS, READONLY, DEBUGGING
 13 .debug_macro  0000504a  00000000  00000000  00028bb3  2**0
                  CONTENTS, READONLY, DEBUGGING
```

For Demo Video Link

https://youtu.be/0h-bhMY6E-M