Benha University

Faculty of Engineering at Shoubra

Communications and Computer Engineering Program (CCEP)

Computer Systems Engineering (CSE)

Anti-Theft System for Automobile

Team Members:

- Abdelaziz Abuzaid Mohamed
- Hazem Youssri Gamal
- Mourad Eldin Nash'at Mourad Muhammed
- Youssef Mamdouh Mohamed El-Nemaky

Supervised by:

Dr. May Mohamed

Supported by:

Valeo

Abstract

Automobile theft became a huge problem that faces every car owner nowadays. As a group of students, we tried to make a proposed solution for this problem to secure cars from theft with an affordable price. Our solution consists of two main processes which are authentication and locating the car. We used three ways for authentication which are RFID module, password using a keypad and face recognition. We used GPS module for locating the car and sending this location by using GSM and the location is saved in ThingSpeak. Our project contains two Tiva C boards and one Raspberry Pi board. The first Tiva C board acts as the control block while the second one acts as the human machine interface block (HMI). The human machine interface connected to RFID, keypad and lcd while the control block is connected to GPS and GSM modules through UART driver, and also connected to external EEPROM with i2c driver. The raspberry pi is connected to a camera to make the face recognition authentication. We also used a mobile application through which the user can lock the engine by disconnecting the electric circuit of the fuel pump if a theft is detected. The mobile application also let the car owner to ignore the authentication process from the application options if they permitted for a friend of him to access the car. The user's data such as their e-mail and password are saved on a firebase. By integrating software drivers such as GPS, GSM, i2c, SPI, UART, keypad, LCD and RFID with the mobile application and face recognition module and integrating hardware components for these modules we can solve the car theft problem and achieve safety for every automobile owner.

Acknowledgment:

We are deeply grateful to our supervisor, Dr. May Mohamed for her guidance, patience, and support. we consider ourselves very fortunate for being able to work with a very considerate and encouraging professor like her. Without her offering to accomplish this research, we would not be able to finish our Graduation project.

We would also like to acknowledge with much appreciation the crucial role of Valeo Egypt who gave us some of the equipment we need and giving us a permission to use all the necessary tools for the whole period of the project, and we are also very grateful to our mentor Tech Lead Eng. Abdelrahman Mabrouk for his guidance and support and trying to answer our questions whenever we asked.

Many thanks go to all lecturers and supervisors who have given their full effort in guiding the team in achieving the goal as well as their encouragement to maintain our progress in track. Our profound thanks to our friends for spending their time in helping and giving support whenever we needed it in fabricating our project.

Contents:

List of Figures:

Chapter 1: Introduction

1.1. Introduction:

Motor vehicle theft (also called car theft and, in the United States, grand theft auto) is the criminal act of stealing or attempting to steal a motor vehicle. In recent years, vehicle thefts are increasing at an alarming rate around the world. Some basic anti-theft systems are used like car alarms installed on your car to stop any thievery attempt. The main problem with car alarms is most of the times when a car produces an alarm, it is a false alarm.

Also, steering wheel locks which are one of the oldest known theft prevention devices on the market today. A steering wheel lock attaches to the steering wheel of the car to lock it in place, preventing someone from driving away. The problems with steering wheels locks are requiring extra key and extra space and Locking mechanism can get jammed during use in colder climates, which is not ideal if you live in frostier temperatures. They may also make a temporary indent in your steering wheel. They might also not fit all vehicles despite universal labeling.

Many People have started to use the technological anti-theft control systems installed in their vehicles. The commercially available anti-theft vehicular systems are very expensive, and therefore we designed our system to be as affordable as possible.

1.2. Problem statement:

Car Thievery is one of the problems that needs addressing in our society, and one that needs to be solved with technological means.

According to studies the average value for car theft in Egypt during the period from 2003 to 2011 was 64 thefts per 100,000 people with a minimum of 42 thefts

per 100,000 people in 2003 and a maximum of 104 thefts per 100,000 people in 2011.

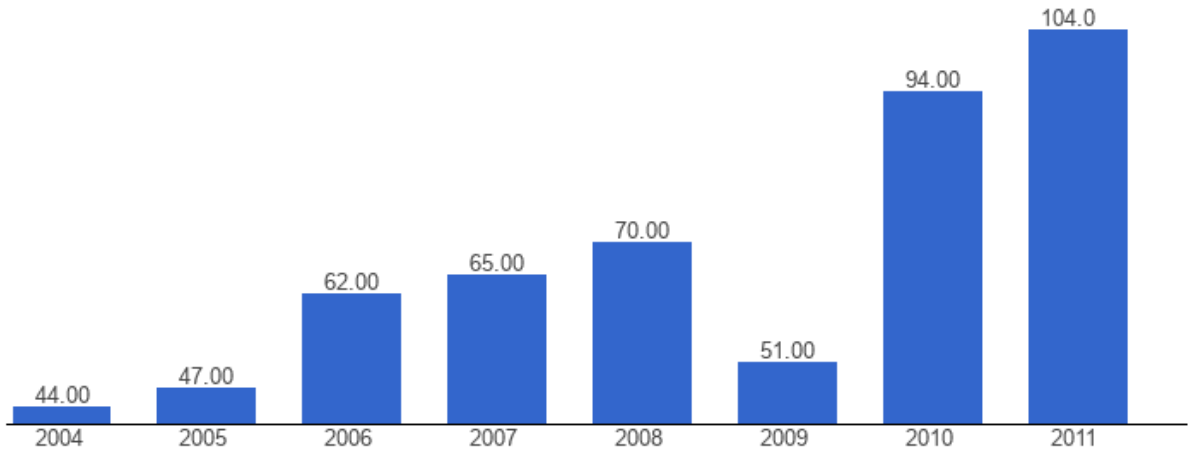


Figure 1

One car stolen in a thousand cars doesn't seem much, now imagine that car is yours!

Between 2015 and 2017, 697,000 motor vehicles were stolen in the EU! And about $6 billion were lost to auto theft in 2018, including the theft of cars, trucks, buses, scooters, snowmobiles, and other road vehicles.

Everyone has the right to feel safe about their properties including their vehicles whether they are expensive cars with tons of options, safety

Year	Vehicles stolen	Percent change
2011	716,508	-3.1%
2012	723,186	0.9
2013	700,288	-3.2
2014	686,803	-1.9
2015	713,063	3.8
2016	767,290	7.6
2017	772,943	0.7
2018	751,904	-2.7
2019	724,872	-3.6
2020	810,400	11.8

and security measures or affordable ones for most of the people. Here is where our project comes in hand.

1.3. Motivation:

For us a group of students who wish to solve this problem, we designed an **Anti-Theft System** that can prevent any unwanted individual from starting the car. The owner -with the needed measures- can use modern technologies that everyone is familiar with, such as **GPS** so you can locate the car whenever you want or use **GSM** to give access to another individual and more! All of this without the need of paying a lot of money to buy a high-class car with all the needed security features that only provided by the manufacturing company.

1.4. Scope:

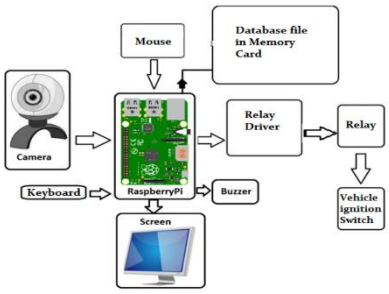
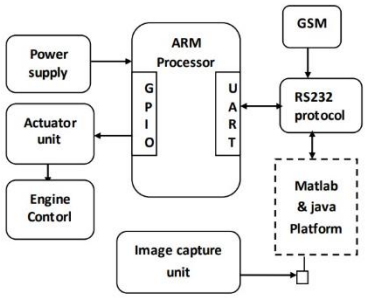
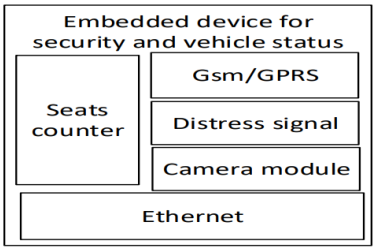
We are targeting both affordable and high end cars that may lack any security or **Anti-Theft systems** with as much low cost as possible, and we are not talking about the need of any manufacturing process we aim to implement this project on an existing car, it's more like an upgrade and we are not talking about cars with specific characteristics or standards or even options, as long the car has a fuel pump! the system can be installed on it.

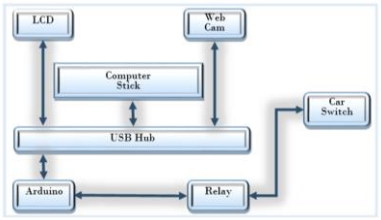
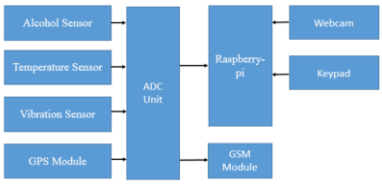
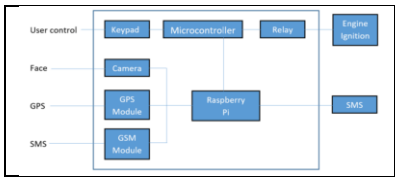
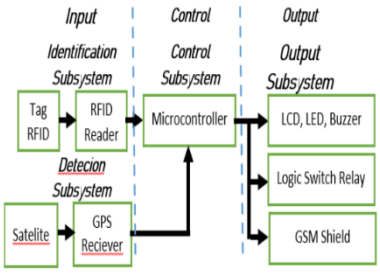
With our device the car should be safe from any attempt to drive, move and in some scenarios even trying to access it to take something from inside.

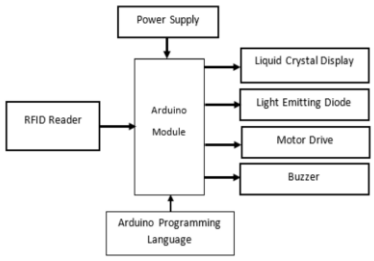
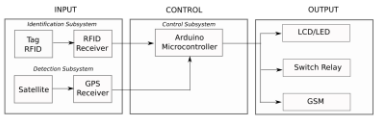
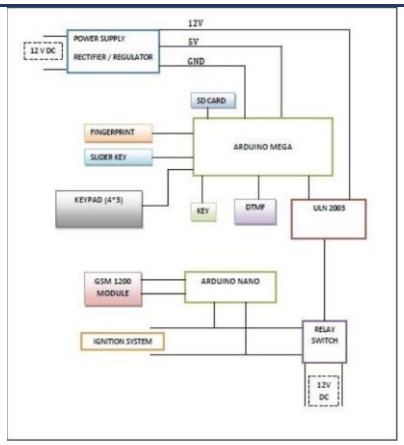
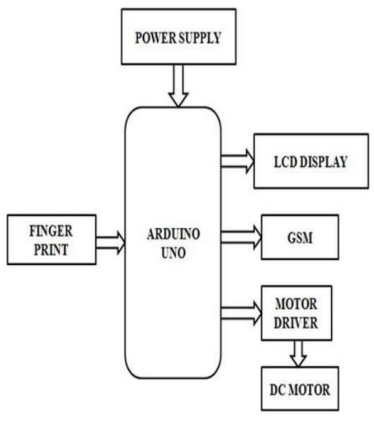
In the next chapters we shall discuss in detail what people achieved in this matter and the strengths and weaknesses of their implementations and designs to the final output then we will proceed to our idea and implementation and how we managed to turn this idea into an actual working device capable of protecting any Automobile from theft.

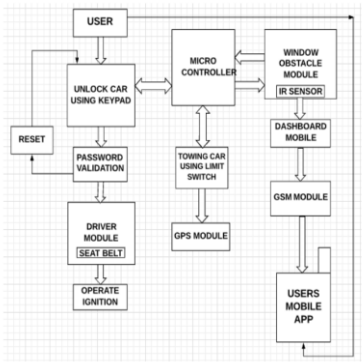
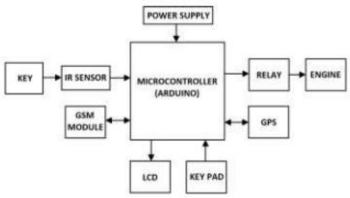
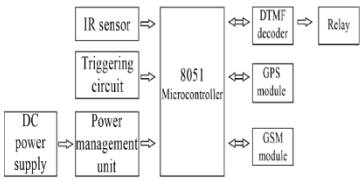
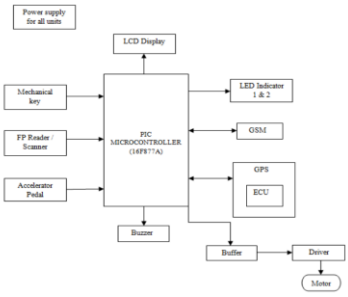
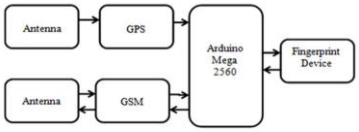
Chapter 2: Previous Work & Research Papers

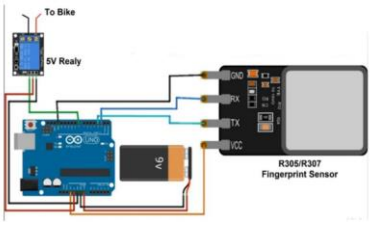
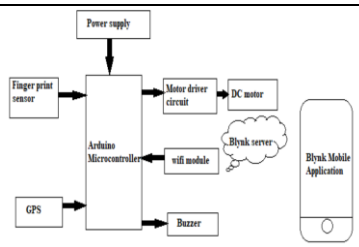
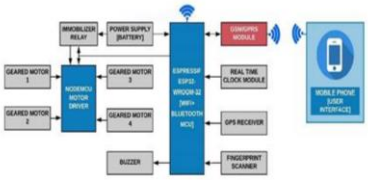
2.1. Previous Work

Technology	Implementation	Strengths	Weaknesses	Block Diagram	Publication Date	Research Name
Face Recognition	Disables vehicle engine automatically.	- Uses facial recognition. - Uses malware detection.	-Facial recognition can be tricked in some instances with high resolution copies of owner.		2021	FACE RECOGNITION BASED INTELLIGENT CAR ANTI-THEFT SYSTEM USING RASPBERRY PI
					2018	Driver Face Recognition: Anti-Theft System
SMS, GSM, GPS, Face Recognition	-Disables vehicle engine automatically. -Detects vehicle theft. -Transmits vehicle coordinates to cellphone.	-Tracking enables recovery of vehicle. -Uses facial recognition. -Uses malware detection.	-GSM messages can be jammed. -Facial recognition can be tricked in some instances with high resolution copies of owner.		2015	Smart Vehicle Security and Defending Against Collaborative Attacks by Malware
					2020	Development of Smart Vehicle Security and Entertainment System (SSES) using Raspberry Pi

					2017	A Cost Effective Method for Automobile Security Based on Detection and Recognition of Human Face
					2018	Face Recognition Based Car Ignition and Security System
					2017	Face Recognition for Motorcycle Engine Ignition with Messaging System
RFID	- Sends coordinates through SMS.	-Immediate notification of user. - Tracking information which can be used to	- Subject to GSM jamming. - Subject to GPS spoofing		2019	Microcontroller-based RFID, GSM and GPS for Motorcycle Security System

		locate vehicle. - Counter Measures to disable stolen vehicle.			2018	Anti-Theft System for Car Security using RFID.
					2018	Smart Security System for Vehicles using Internet of Things (IoT)
Biometrics (Only Fingerprint)	- Uses fingerprint to authenticate user. - Uses password as a backup system.	- Biometrics enhance the traditional security. - Password provides contingency	- Possibly susceptible to fingerprint spoofing. - Access to password by attacker will enable biometrics bypass.		2018	ANTI-THEFT VEHICLE SECURITY SYSTEM USING FINGERPRINT SCANNER AS WELL AS MANUAL
					2020	Fingerprint based Anti Theft for Two Wheelers Authentication of Vehicle users

GPS, GSM, keypad	Uses password	Provides tracking capabilities.	Security can be bypassed with just a key.		2018	An Attempt to Develop an IOT based Vehicle Security System
					2018	Anti-Theft Security System for Vehicles
IR Sensor, GPS, GSM	- Detects motion and calls the owner. - Owner can cut off fuel supply.	- Provides tracking capabilities. - Counter Measures to protect the car from being stolen.	- Subject to GSM jamming. - Subject to GPS spoofing		2015	Advanced Vehicle Security System
GPS, GSM, Fingerprint	- Uses fingerprint to authenticate user. - Uses password as a backup system. - Transmits vehicle	- Biometrics enhance the traditional security. - Password provides contingency	- Possibly susceptible to fingerprint spoofing. - Access to password by attacker will enable biometrics bypass.		2015	Real Time Biometrics based Vehicle Security System with GPS and GSM Technology
					2017	Anti-theft Protection of Vehicle by GSM & GPS with Fingerprint Verification

	coordinates to cellphone. -Owner can cut off fuel supply.		- Subject to GSM jamming. - Subject to GPS spoofing		2019	Anti-theft Fingerprint Security System for Motorcycles Using Arduino UNO, GPS/GSM Module
					2020	Fingerprint Authentication and Mobile App Based Monitoring of Vehicles Using IOT
					2020	Development of an Anti-Theft Vehicle Security System using GPS and GSM Technology with Biometric Authentication

2.2. Conclusion

There are strengths and weaknesses in every technology applied, but that's just how it is, nothing is perfect. That's why our team has learnt and used these

technologies in our system taking the strengths and trying to eliminate the weaknesses.

Our proposed system will have:

- GPS: to locate the vehicle
- GSM: to communicate with the system remotely
- Three-way authentication:
    1. RFID
    2. Face recognition
    3. Passcode

  These features are used in a certain pattern to assure only the owner will start the vehicle.
- Camera installed can work both in face recognition software and taking pictures and send them to the user via the mobile application
- Mobile application: connected with the system installed in the vehicle to facilitate the access of some features of the system, as well as give access to other individuals

Chapter 3: Proposed Solution

The first part and probably the most important, the largest part and core of our project is the concepts and implementation of embedded systems using different software drivers and hardware components but first we need to know which software and hardware pieces were used to develop this system:

3.1 Software Background:

Software development is the initial step in developing any application not just embedded systems as it defines in which way the hardware will be implemented and also it considered the testing phase as errors will be easy to handle and we have the following software drivers used in this table:

MCAL	HAL
PORT (AUTOSAR)	LCD
DIO (AUTOSAR)	Keypad
UART	GSM Module
Timer	GPS Module
SPI	RFID Module
I2C	
External Interrupt	
External EEPROM	

Basic Software Layered Architecture in Embedded Systems

Is the process of dividing software into layers. There are 3 main layers in embedded systems as shown in Fig

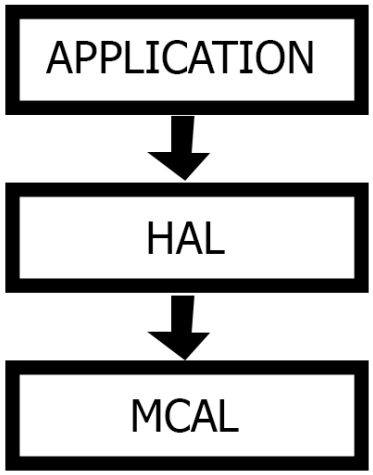


Figure 3. 1

Application layer is mostly written in high level languages like C++, C# with rich GUI support. Application layer calls the middleware api in response to action by the user or an event. Mostly all logic is done in the application layer.

HAL (Hardware abstraction) layer is the middleware software that maintains the state machine of the device. And is responsible to handle requests from the upper layer and the lower-level layer. The middleware exposes a set of api functions which the application must call to use the services offered by the middleware. And vice versa the middleware can send data to the application layer via IPC mechanism.

MCAL (Microcontroller abstraction layer): is responsible for talking to the chipset either configuring registers or reading from the chipset registers. The firmware exposes a set of api's that the middleware can call. In order to perform specific tasks.

Advantages of dividing software in embedded systems into layers:

- Isolate application from hardware changes as upper layers don't be affected by lower ones (Portability: the ability to port the code into other hardware modules without changing it or with a minimum number of changes as possible)
- Provides the opportunity for re-use by defining software building blocks.
- Simplifies isolation of software bugs, easier in testing (detection and correction)
- Facilitates software maintainability.

Disadvantages of dividing software in embedded systems into layers:

- May incur slight to moderate processing overhead. More processing time due to calls between different layers and different modules inside each layer.
- Requires more hardware resources (Ram, storage, processing power, … etc.).

As we can see the software code is divided into three main parts MCAL, HAL and APP Layers, all these layers work together to provide abstraction for the user and in case of errors you will only need fix one layer not all of them and Eases porting from one platform to another platform.

AUTOSAR:

Automotive trends like autonomous driving, v2x connectivity, OTA updates, predictive maintenance, and many other innovative features are based on in-vehicle software functions. For all these functions to work seamlessly and to cater

to real-time in-vehicle functionalities, each ECU must work efficiently. Modern-day high-end vehicles have up to 100 ECUs, which communicate with each other via CAN bus, CAN FD, or Ethernet network to support complex vehicular functions.

Earlier, ECU software used by OEMs was on different platforms. There was no standard software architecture that was being used by tier 1 suppliers and their vendors to design the ECU software for OEMs. So, whenever any OEM wishes to switch to a new tier 1 supplier or vice-versa, the transition was very difficult. The new supplier used to face enormous challenges in understanding the existing software architecture, hardware platforms, and standards used in ECU software development. Thus, it was nearly impossible for a new supplier to drive an on-going project from the midst of its production life cycle.

To streamline the coordination between OEMs and tier 1 suppliers, to improve ECU software quality and reduce development time and costs, tier 1 automotive suppliers, semiconductor manufacturers, software suppliers, tool suppliers, and others came forward in 2003 and created a consortium called **Aut**omotive **O**pen **S**ystem **Ar**chitecture (**AUTOSAR**).

What is AUTOSAR?

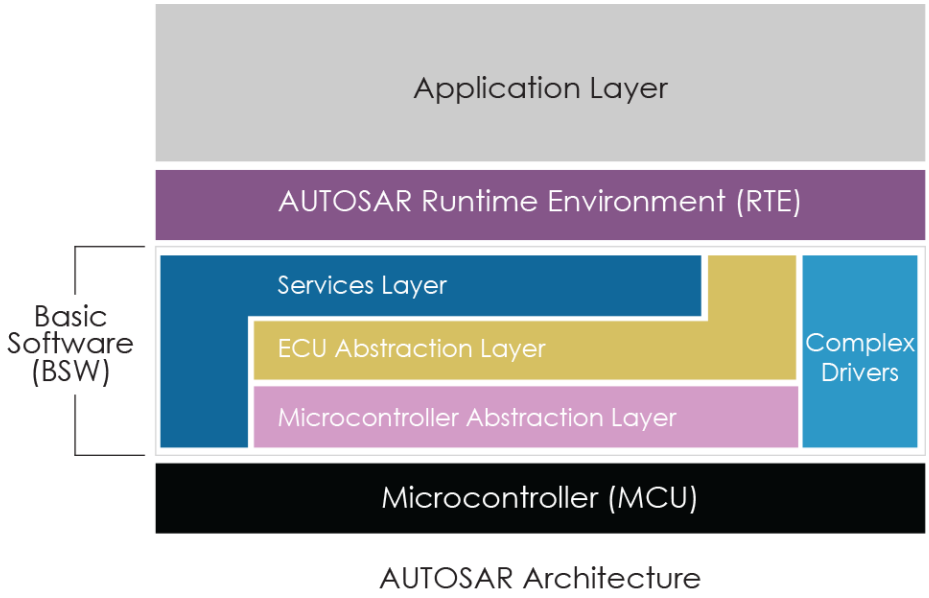
Automotive Open System Architecture (AUTOSAR) is an open and standardized automotive software architecture, which supports standardization in interfaces between application software and basic vehicular functions and it helps in establishing common ECU software architecture for all the AUTOSAR members.

AUTOSAR is intended to provide inherent benefits to the members to manage increasingly complex E/E in-vehicle environments like easy integration and exchange of functions in complex ECU network and control over the entire product lifecycle.

AUTOSAR Architecture:

AUTOSAR is an open system architecture for automotive software development and provides standards for developing common automotive software applications. It is a growing and evolving standard that defines a layered architecture for the software. The classic AUTOSAR platform runs on a microcontroller and is divided into 3 main layers; let us discuss them in detail:

- Basic Software Architecture- It is common to any AUTOSAR ECU.
- AUTOSAR Runtime Environment
- Application Layer



AUTOSAR Architecture

Figure 3. 2

Basic Software Architecture (BSW):

AUTOSAR Basic Software Architecture consists of hundreds of software modules structured in different layers and is common to any AUTOSAR ECU. This means the supplier who has designed BSW can share it with other suppliers that are working on ECUs of engine, gearbox, etc.

- ### Microcontroller Abstraction Layer (MCAL):

  MCAL is also known as a hardware abstraction layer and implements interface for the specific microcontroller. MCAL has layers of software, which are integrated with the microcontroller through registers, and offers drivers like system drivers, diagnostics drivers, memory drivers, communication drivers (CAN, LIN, Ethernet, etc.), I/O drivers and more.

- ### ECU Abstraction Layer:

  The prime task of the ECU abstraction layer is to deliver higher software layers ECU specific services. This layer and its drivers are independent of the microcontroller and dependent on the ECU hardware and provide access to all the peripherals and devices of ECU, which supports functionalities like communication, memory, I/O, etc.

- ### Service Layer:

  The service layer is the topmost layer of AUTOSAR Basic Software Architecture. The service layer constitutes an operating system, which runs from the application layer to the microcontroller at the bottom. The OS has an interface between the microcontroller and the application layer and can schedule application tasks. The service

layer in BSW is responsible for services like network services, memory services, diagnostics service, communication service, ECU state management, and more.

AUTOSAR Runtime Environment (RTE Layer):

AUTOSAR Run-time Environment is a middleware layer of the AUTOSAR software architecture between the BSW and the application layer and provides communication services for the application software.

Application Layer:

The application layer is the first layer of the AUTOSAR software architecture and supports custom functionalities implementation. This layer consists of the specific software components and many applications which perform specific tasks as per instructions.

The AUTOSAR application layer consists of three components which are: **application software components, ports of software components, and port interfaces.**

AUTOSAR ensures standardized interfaces for software components in the application layer and application software components help in generating simple applications to support the vehicle functions.

The communication between software components is enabled via specific ports using a virtual Function Bus. These ports also facilitate communication between software components and AUTOSAR Basic Software (BSW).

The above-explained architecture of AUTOSAR is its classic platform, which supports real-time requirements and safety constraints. Based on the microcontroller, the classic platform is capable of supporting applications in the field of networking and security by allowing ECUs to access vehicle sensors and actuators.

Why Adaptive AUTOSAR?

From 2003 to 2015, Classic AUTOSAR became an established platform and was doing perfectly well to run 60-80 ECUs in a vehicle. With the evolution of IoT based automotive trends like V2X connectivity and automated driving, electrification skyrocketed, and as a result, a huge demand for supporting functions and devices was created in the market. It was discovered that the existing classic AUTOSAR platform was not suitable to support these mega-trends and new architecture with more powerful and flexible E/E architecture is required. Adaptive AUTOSAR, the new architecture was released to support these functions and 1st release of AUTOSAR adaptive platform was done in March 2017.

Adaptive AUTOSAR architecture comes with a central application server, which assists high-performance computing. Ethernet-based ECUs in this system supports real-time functionalities. Adaptive AUTOSAR is scalable and has dynamic architecture, in which applications can be updated over the vehicle's lifecycle. It enables OEMs to deploy high-tech software features in a vehicle and update them over-the-air whenever required.

AUTOSAR adaptive architecture supports all the futuristic automotive applications like infotainment, v2x, predictive maintenance, automotive apps, ADAS functions with a camera, RADAR and LIDAR sensors, map updates, electrification, and more.

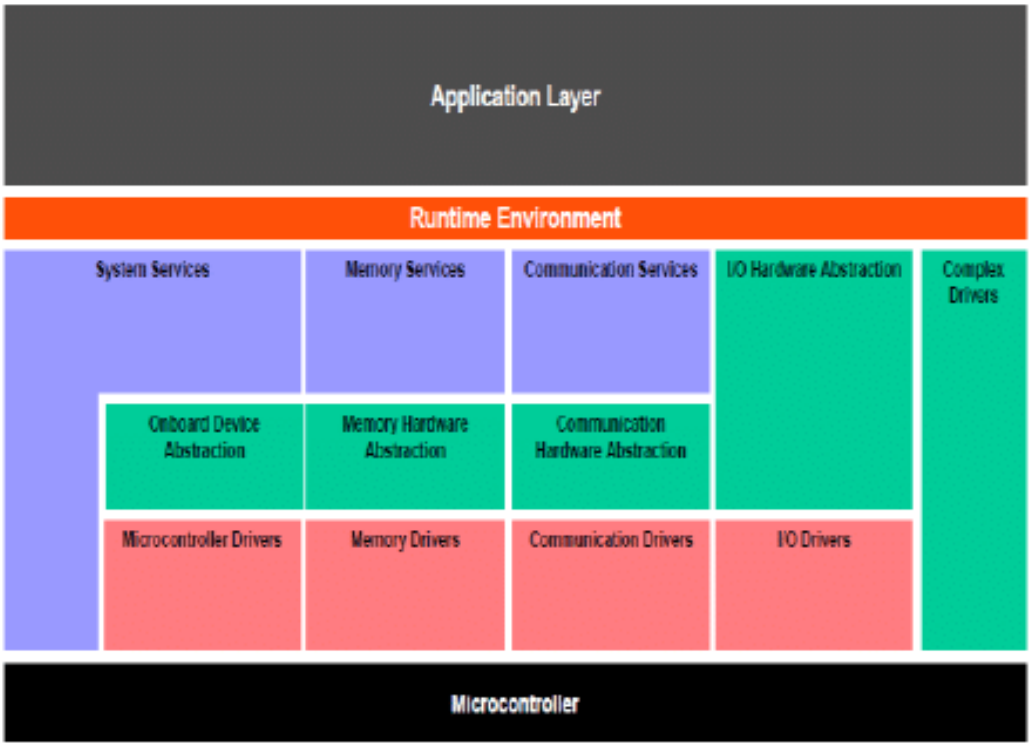


Figure 3. 3

DIO Driver:

The DIO driver provides port and channel based read and write access to the internal general purpose I/O ports. The read and write behavior is unbuffered. The basic behavior of this driver is synchronous.

Some of the important functions used in the DIO Drivers are

Function	Dio_Init	
Syntax	void Dio_Init(const Dio_ConfigType* ConfigPtr)	
Parameters	ConfigPtr	Pointer to post-build configuration data

Return Value	None
Description	Initializes the module.

Function	Dio_ReadChannel	
Syntax	Dio_LevelType Dio_ReadChannel(Dio_ChannelType ChannelId)	
Parameters	ChannelId	ID of DIO channel
Return Value	Dio_LevelType	STD_HIGH STD_LOW
Description	Service to read a level of a channel.	

Function	Dio_WriteChannel	
Syntax	void Dio_WriteChannel(Dio_ChannelType ChannelId, Dio_LevelType Level)	
Parameters	ChannelId	ID of Dio Channel
	Level	Value to be written
Return Value	None	
Description	Service to set a level of a channel.	

Function	Dio_ReadPort
Syntax	Dio_PortLevelType Dio_ReadPort(Dio_PortType PortId)

Parameters	PortId	ID of DIO port
Return Value	Dio_PortLevelType	Level of all channels of that port
Description	Returns the level of all channels of that port	

Function	Dio_WritePort	
Syntax	void Dio_WritePort(Dio_PortType PortId, Dio_PortLevelType Level)	
Parameters	PortId	ID of DIO port
	Level	Value to be written
Return Value	None	
Description	service to set a value of the port.	

Function	Dio_ReadChannelGroup	
Syntax	Dio_PortLevelType Dio_ReadChannelGroup(const Dio_ChannelGroupType* ChannelGroupIdPtr)	
Parameters	ChannelGroupIdPtr	Pointer to ChannelGroup
Return Value	Dio_PortLevelType	Level of a subset of the adjoining bits of a port
Description	This Service reads a subset of the adjoining bits of a port.	

Function	Dio_WriteChannelGroup

Syntax	void Dio_WriteChannelGroup(const Dio_ChannelGroupType* ChannelGroupIdPtr, Dio_PortLevelType Level)	
Parameters	ChannelGroupIdPtr	Pointer to ChannelGroup
	Level	Value to be written
Return Value	None	
Description	Service to set a subset of the adjoining bits of a port to a specified level.	

Port Driver

This module initializes the whole port structure of the microcontroller. Many ports and port pins can be assigned to various functionalities like e.g.

- General purpose I/O
- ADC
- SPI
- SCI
- PWM

For this reason, there has to be an overall configuration and initialization of this port structure. The configuration and usage of those port pins is microcontroller and ECU dependent.

Some of the important functions used in Port Driver are:

Function	Port_Init	
Syntax	void Port_Init(const Port_ConfigType* ConfigPtr)	
Parameters	ConfigPtr	Pointer to configuration set.
Return Value	None	
Description	Initializes the Port Driver module.	

Function	Port_RefreshPortDirection
Syntax	void Port_RefreshPortDirection(void)
Parameters	None
Return Value	None
Description	Refreshes port direction.

Function	Port_SetPinDirection	
Syntax	void Port_SetPinDirection(Port_PinType Pin, Port_PinDirectionType Direction)	
Parameters	Pin	Port Pin ID number
	Direction	Port Pin Direction
Return Value	None	
Description	Sets the port pin direction	

Function	Port_SetPinMode	
Syntax	void Port_SetPinMode(Port_PinType Pin, Port_PinModeType Mode)	
Parameters	Pin	Port Pin ID number
	Mode	New Port Pin mode to be set on port pin.
Return Value	None	
Description	Sets the port pin mode.	

Communication Drivers used in our project

UART – Universal Asynchronous Receiver/Transmitter Driver

Responsible for communication between the main components of the system e.g.

- Control block and human machine interface (HMI)
- Control block and raspberry pi

It is also used for communication between

- control block and gsm/gps module.

Some of the important functions used in UART driver are:

Function	Uart_Init	
Syntax	void Uart_Init(const Uart_ConfigType * ConfigPtr)	
Parameters	ConfigPtr	Pointer to configuration set

Return Value	None	
Description	Initializes the UART driver module.	

Function	Uart_SendByte	
Syntax	void Uart_SendByte(Uart_ModuleNumber uartModuleNumber ,const uint8 byteToSend)	
Parameters	uartModuleNumber	ID of the used uart module
	byteToSend	Data required to be sent
Return Value	None	
Description	Send byte to another device	

Function	Uart_ReceiveByte	
Syntax	uint8 Uart_ReceiveByte(Uart_ModuleNumber uartModuleNumber)	
Parameters	uartModuleNumber	ID of the used uart module

Return Value	None
Description	Receive byte from another device

I2C - Inter-Integrated Circuit Driver

Responsible for communication between the control block and EEPROM where the master is the control block and the EEPROM is the slave.

Some of the important functions used in I2C driver are:

Function	I2c_Init	
Syntax	void I2c_Init(const I2c_ConfigType * ConfigPtr)	
Parameters	ConfigPtr	Pointer to configuration set
Return Value	None	
Description	Initializes the I2C driver module.	

Function	I2c_WriteByte
Syntax	uint8 I2c_WriteByte(I2c_ModuleNumber i2cModuleNumber,uint8 slaveAddress, uint8 slaveMemoryAddress, uint8 byte)

Parameters	i2cModuleNumber	ID of the used I2C module
	slaveAddress	The address of the slave
	slaveMemoryAddress	The address of the internal register of the slave
	byte	The byte to write to the internal register of the slave
Return Value	Uint8 (unsigned 8-bit integer)	Error code 0 if no errors
Description	Writes a byte to the slave device	

Function	I2c_ReadByte	
Syntax	uint8 I2c_ReadByte(I2c_ModuleNumber i2cModuleNumber,uint8 slaveAddress, uint8 slaveMemoryAddress,uint8 * data)	
Parameters	i2cModuleNumber	ID of the used I2C module
	slaveAddress	The address of the slave
	slaveMemoryAddress	The address of the internal register of the slave
	data	A pointer to a variable which will hold the returned value.
Return Value	Uint8 (unsigned 8-bit integer)	Error code 0 if no errors
Description	Reads a byte from the slave device	

SPI – Serial Peripheral Interface Driver

Responsible for the communication between the human machine interface (HMI) and RFID reader.

Some of the important functions used in SPI driver are:

Function	Spi_Init	
Syntax	void Spi_Init(const Spi_ConfigType * ConfigPtr);	
Parameters	ConfigPtr	Pointer to configuration set
Return Value	None	
Description	Initializes the SPI driver module.	

Function	SPI_SendReceiveByte	
Syntax	uint8 SPI_SendReceiveByte(SSI_ModuleNumber moduleNumber,uint8 data);	
Parameters	moduleNumber	ID of the used SPI module
	data	Data to send

Return Value	Uint8 (unsigned 8-bit integer)	Received data
Description	Send the required data through SPI to the other SPI device. In the same time data will be received from the other device	

GPT – General Purpose Timer Driver

Almost every project in embedded systems has a timer driver used in it.

The timer driver gives the ability to time events. We have used timers widely in our project.

- Locking the system for specific interval of time when a failed authentication occurs multiple times.
- Reporting the location of the car to the server.
- Giving the user specific time to restart the vehicle after switching off before the system locks the fuel pump again.

Some of the important functions used in GPT driver are:

Function	Timer_Init
Syntax	void Timer_Init(void)
Parameters	None

Return Value	None
Description	Initializes the timer driver module.

Function	Timer_Start
Syntax	void Timer_Start(void)
Parameters	None
Return Value	None
Description	Starts the timer module.

Function	Timer_Stop
Syntax	void Timer_Stop(void);
Parameters	None
Return Value	None
Description	Stops the timer module.

UART

UART Stands for:

(Universal asynchronous receiver-transmitter)

UART Meaning

It is a communication protocol which is one of the simplest and oldest forms of device-to-device digital communication. we can find UART devices as a part of integrated circuits (ICs) or as individual components. UARTs communicate between two separate nodes using a pair of wires and a common ground.

In a UART communication scheme:

1. One chip's Tx (transmit) pin connects directly to the other's Rx (receive) pin and vice versa. Commonly, the transmission will take place at 3.3 or 5V. UART is a single-master, single-slave protocol, (where one device is set up to communicate with only one partner).

2. When sending on the Tx pin, the first UART translates this parallel information into serial and transmits it to the receiving counterpart.

3. The second UART receives this data on its Rx pin and transforms it back into parallel to communicate with its controlling device.

UARTs transmit data serially, in one of three modes:

- **Full duplex**: Simultaneous communication to and from each master and slave

- **Half duplex**: Data flows in one direction at a time

- **Simplex**: One-way communication only

UART Frame

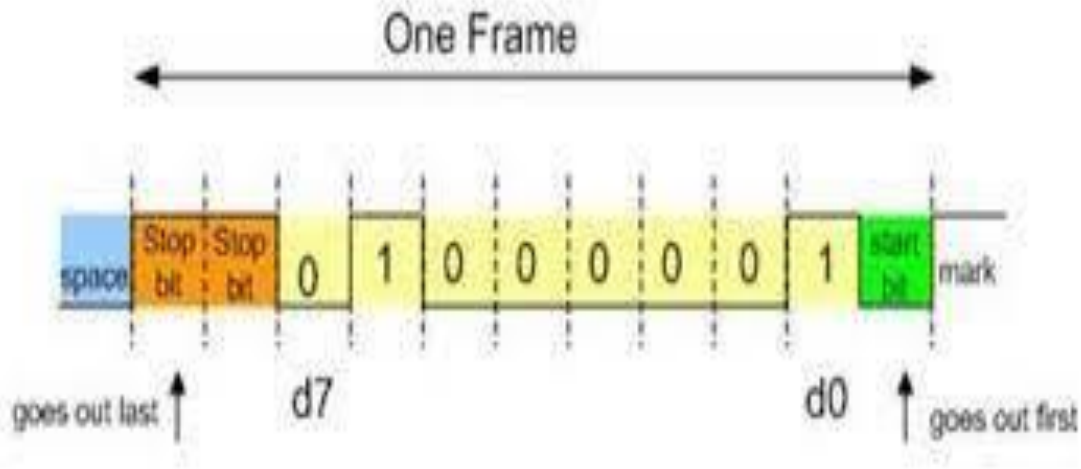


Figure 3. 4

Start Bit

The UART data transmission line is normally held at a high voltage level when it's not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one (1) clock cycle. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate.

Data Frame

The data frame contains the actual data being transferred. It can be five (5) bits up to eight (8) bits long if a parity bit is used. If no parity bit is used, the data frame can be nine (9) bits long. In most cases, the data is sent with the least significant bit first.

Parity

Parity describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission. Bits can be changed by electromagnetic radiation, mismatched baud rates, or long-distance data transfers.

After the receiving UART reads the data frame, it counts the number of bits with a value of 1 and checks if the total is an even or odd number. If the parity bit is a 0 (even parity), the 1 or logic-high bit in the data frame should total to an even number. If the parity bit is a 1 (odd parity), the 1 bit or logic highs in the data frame should total to an odd number.

When the parity bit matches the data, the UART knows that the transmission was free of errors. But if the parity bit is a 0, and the total is odd, or the parity bit is a 1, and the total is even, the UART knows that bits in the data frame have changed.

Stop Bits

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for one (1) to two (2) bit(s) duration.

Uses of UART in our Project

1. Communication between control block and HMI block.
2. Communication between control block and Raspberry pi.
3. RFID.

SPI

Serial peripheral interface (SPI) is one of the most widely used interfaces between microcontroller and peripheral ICs such as sensors, ADCs, DACs, shift registers, SRAM, and others. We are providing a brief description of the SPI interface followed by an introduction to Analog Devices' SPI enabled switches and muxes, and how they help reduce the number of digital GPIOs in system board design.

SPI is a synchronous, full duplex main-subnode-based interface. The data from the main or the subnode is synchronized on the rising or falling clock edge. Both main

and subnode can transmit data at the same time. The SPI interface can be either 3-wire or 4-wire. We simply focus on the popular 4-wire SPI interface.

Interface

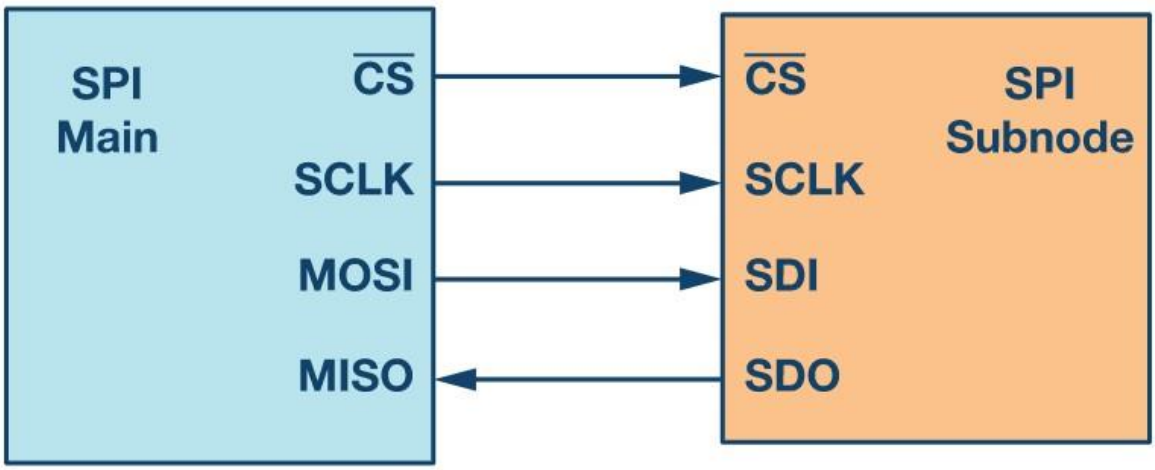


Figure 3. 5

Figure 1. SPI configuration with main and a subnode.

4-wire SPI devices have four signals:

- Clock (SPI CLK, SCLK)
- Chip select (CS)
- main out, subnode in (MOSI)
- main in, subnode out (MISO)

The device that generates the clock signal is called the main. Data transmitted between the main and the subnode is synchronized to the clock generated by the main. SPI devices support much higher clock frequencies compared to I^2C interfaces. Users should consult the product data sheet for the clock frequency specification of the SPI interface.

SPI interfaces can have only one main and can have one or multiple subnodes. Figure 1 shows the SPI connection between the main and the subnode.

The chip select signal from the main is used to select the sub node. This is normally an active low signal and is pulled high to disconnect the sub node from the SPI bus. When multiple sub nodes are used, an individual chip select signal for each sub node is required from the main. In this here, the chip select signal is always an active low signal.

MOSI and MISO are the data lines. MOSI transmits data from the main to the sub node and MISO transmits data from the sub node to the main.

Data Transmission

To begin SPI communication, the main must send the clock signal and select the sub node by enabling the CS signal. Usually, chip select is an active low signal; hence, the main must send a logic 0 on this signal to select the sub node. SPI is a full-duplex interface; both main and sub node can send data at the same time via the MOSI and MISO lines respectively. During SPI communication, the data is simultaneously transmitted (shifted out serially onto the MOSI/SDO bus) and received (the data on the bus (MISO/SDI) is sampled or read in). The serial clock edge synchronizes the shifting and sampling of the data. The SPI interface provides the user with flexibility to select the rising or falling edge of the clock to sample and/or shift the data. Please refer to the device data sheet to determine the number of data bits transmitted using the SPI interface.

Clock Polarity and Clock Phase

In SPI, the main can select the clock polarity and clock phase. The CPOL bit sets the polarity of the clock signal during the idle state. The idle state is defined as the period when CS is high and transitioning to low at the start of the transmission and when CS is low and transitioning to high at the end of the transmission. The CPHA bit selects the clock phase. Depending on the CPHA bit, the rising or falling clock edge is used to sample and/or shift the data. The main must select the clock polarity and clock phase, as per the requirement of the subnode. Depending on the CPOL and CPHA bit selection, four SPI modes are available. Table 1 shows the four SPI modes.

SPI Mode	CPOL	CPHA	Clock Polarity in Idle State	Clock Phase Used to Sample and/or Shift the Data
0	0	0	Logic low	Data sampled on rising edge and shifted out on the falling edge
1	0	1	Logic low	Data sampled on the falling edge and shifted out on the rising edge
2	1	0	Logic high	Data sampled on the rising edge and shifted out on the falling edge
3	1	1	Logic high	Data sampled on the falling edge and shifted out on the rising edge

Table 1. SPI Modes with CPOL and CPHA

Figure 2 through Figure 5 show an example of communication in four SPI modes. In these examples, the data is shown on the MOSI and MISO line. The start and end of transmission is indicated by the dotted green line, the sampling edge is indicated in orange, and the shifting edge is indicated in blue. Please note these figures are for illustration purpose only. For successful SPI communications, users must refer to the product data sheet and ensure that the timing specifications for the part are met.

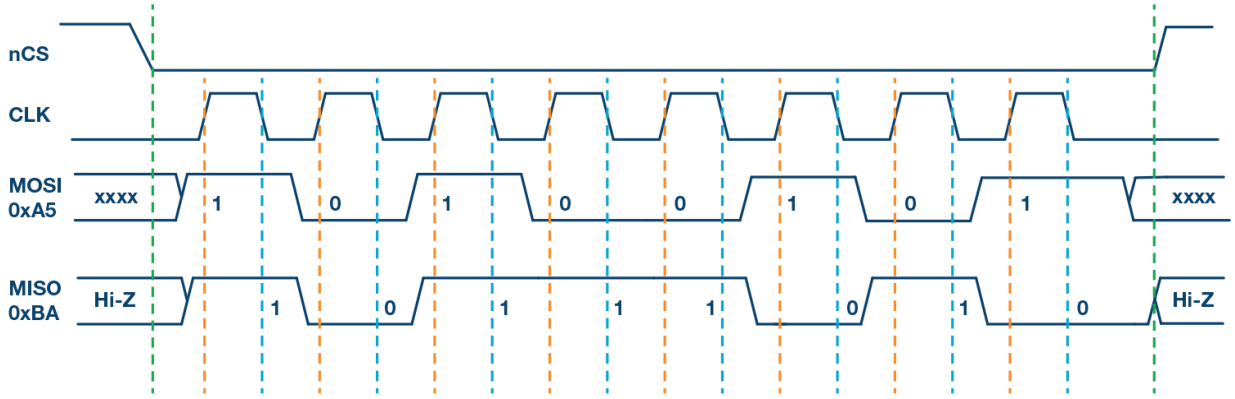


Figure 2. SPI Mode 0, CPOL = 0, CPHA = 0: CLK idle state = low, data sampled on rising edge and shifted on falling edge.

Figure 3 shows the timing diagram for SPI Mode 1. In this mode, clock polarity is 0, which indicates that the idle state of the clock signal is low. The clock phase in this mode is 1, which indicates that the data is sampled on the falling edge (shown by the orange dotted line) and the data is shifted on the rising edge (shown by the dotted blue line) of the clock signal.

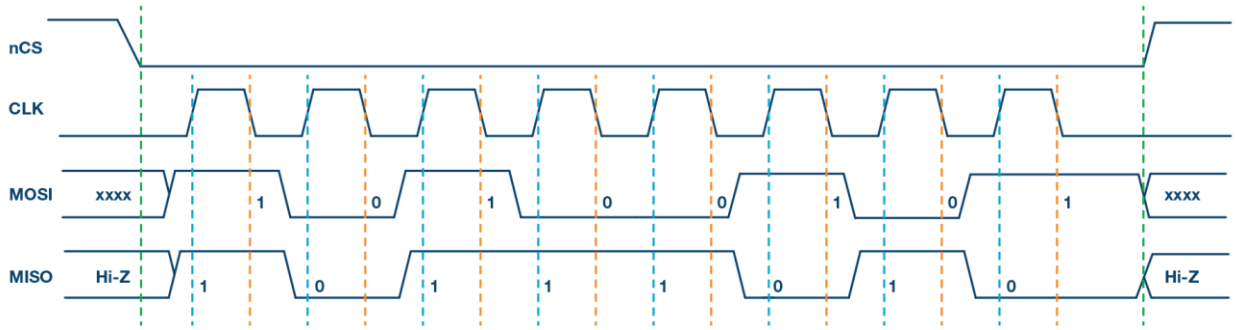


Figure 3. SPI Mode 1, CPOL = 0, CPHA = 1: CLK idle state = low, data sampled on the falling edge and shifted on the rising edge.

Figure 4 shows the timing diagram for SPI Mode 3. In this mode, the clock polarity is 1, which indicates that the idle state of the clock signal is high. The clock phase in this mode is 1, which indicates that the data is sampled on the falling edge (shown by the orange dotted line) and the data is shifted on the rising edge (shown by the dotted blue line) of the clock signal.

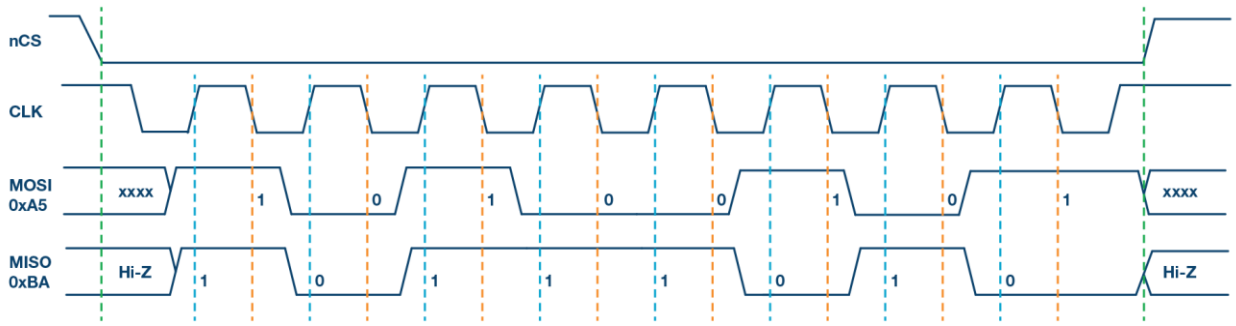


Figure 4. SPI Mode 3, CPOL = 1, CPHA = 1: CLK idle state = high, data sampled on the falling edge and shifted on the rising edge.

Figure 5 shows the timing diagram for SPI Mode 2. In this mode, the clock polarity is 1, which indicates that the idle state of the clock signal is high. The clock phase in this mode is 0, which indicates that the data is sampled on the rising edge (shown by the orange dotted line) and the data is shifted on the falling edge (shown by the dotted blue line) of the clock signal.

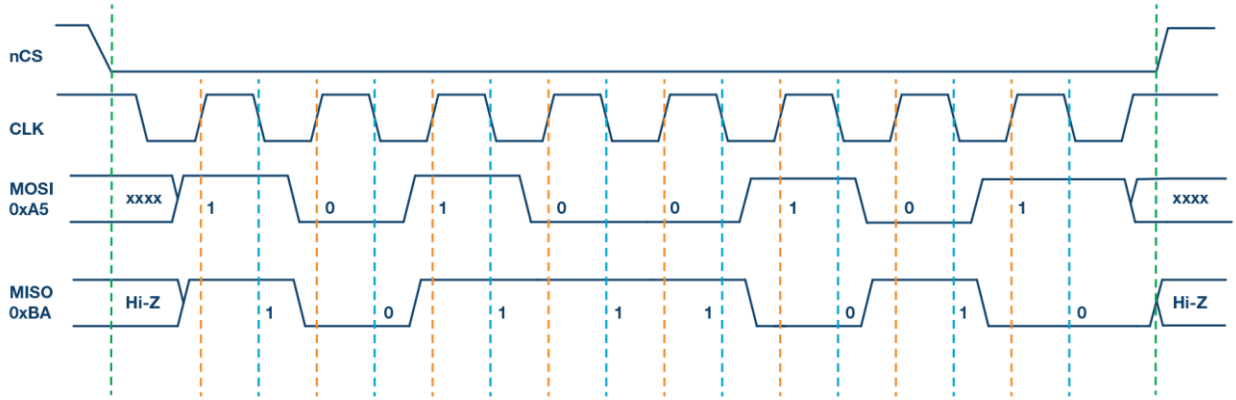


Figure 5. SPI Mode 2, CPOL = 1, CPHA = 0: CLK idle state = high, data sampled on the rising edge and shifted on the falling edge.

SPI in our project

We use SPI to communicate with the RFID peripheral, as it is used in the authentication process.

I2C

stands for **Inter-Integrated Circuit.** It is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it is a widely used protocol for short-distance communication. It is also known as Two Wired Interface (TWI).

Working of I2C communication protocol:

It uses only 2 bi-directional open-drain lines for data communication called SDA and SCL. Both these lines are pulled high.

Serial Data (SDA) – Transfer of data takes place through this pin.
Serial Clock (SCL) – It carries the clock signal.
I2C operates in 2 modes:

- Master mode
- Slave mode

Each data bit transferred on SDA line is synchronized by a high to the low pulse of each clock on the SCL line.

According to I2C protocols, the data line cannot change when the clock line is high, it can change only when the clock line is low. The 2 lines are open drain; hence a pull-up resistor is required so that the lines are high since the devices on the I2C bus are active low. The data is transmitted in the form of packets which comprises 9 bits. The sequence of these bits is:

1. Start Condition 1 bit
2. Slave Address 8 bit
3. Acknowledge 1 bit

Start and Stop Conditions:

Start and stop can be generated by keeping the SCL line high and changing the level of SDA. To generate start, condition the SDA is changed from high to low while keeping the SCL high. To generate STOP condition SDA goes from low to high while keeping the SCL high.

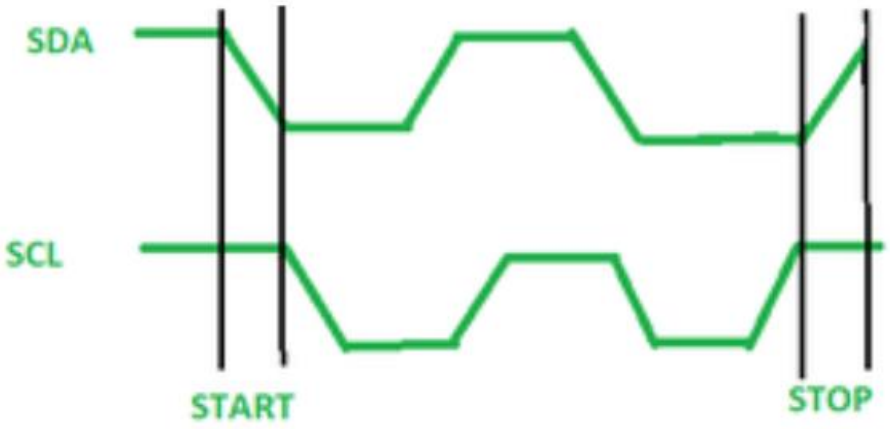


Figure 3. 6

Repeated Start Condition:

Between each start and stop condition pair, the bus is considered as busy and no master can take control of the bus. If the master tries to initiate a new transfer and does not want to release the bus before starting the new transfer, it issues a new start condition. It is called a repeated start condition.

Read/Write Bit:

A high Read/Write bit indicates that the master is sending the data to the slave, whereas a low Read/Write bit indicates that the master is receiving data from the slave.

ACK/NACK Bit:

After every data frame, follows an ACK/NACK bit. If the data frame is received successfully then ACK bit is sent to the sender by the receiver.

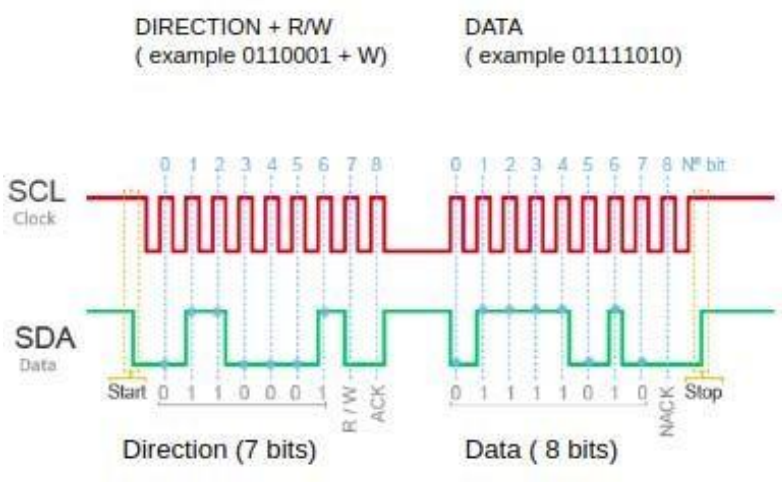


Figure 3. 7

Addressing:

The address frame is the first frame after the start bit. The address of the slave with which the master wants to communicate is sent by the master to every slave connected with it. The slave then compares its own address with this address and sends ACK.

I2C Packet Format:

In the I2C communication protocol, the data is transmitted in the form of packets. These packets are 9 bits long, out of which the first 8 bits are put in SDA line and the 9th bit is reserved for ACK/NACK i.e. Acknowledge or Not Acknowledge by the receiver.

Start condition plus **address packet** plus one more **data packet** plus **stop condition** collectively form a complete **data transfer.**

Features of I2C communication protocol:

- **Half-duplex communication protocol:**
  Bi-directional communication is possible but not simultaneously.

- **Synchronous communication:**
  The data is transferred in the form of frames or blocks.
  Can be configured in a multi-master configuration.

- **Clock Stretching:**
  The clock is stretched when the slave device is not ready to accept more data by holding the SCL line low, hence disabling the master to raise the clock line. Master will not be able to raise the clock line because the wires are AND wired and wait until the slave releases the SCL line to show it is ready to transfer next bit.

- **Arbitration:**
  I2C protocol supports multi-master bus system but more than one bus cannot be used simultaneously. The SDA and SCL are monitored by the masters. If the SDA is found high when it was supposed to be low it will be inferred that another master is active and hence it stops the transfer of data.

- **Serial transmission:**
  I2C uses serial transmission for transmission of data.
  Used for low-speed communication.

Advantages:

- Can be configured in multi-master mode.
- Complexity is reduced because it uses only 2 bi-directional lines (unlike SPI Communication).
- Cost-efficient.
- It uses ACK/NACK feature due to which it has improved error handling capabilities.

Limitations:

- Slower speed.
- Half-duplex communication is used in the I2C communication protocol.

I2C in our project

I2C is used to read and write from and to the EEPROM, which is used to store some data permanently

Interrupts:

Introduction:

An **interrupt** is a condition that causes the microprocessor to temporarily work on a different task, and then later return to its previous task. Interrupts can be internal or external. Internal interrupts, or "software interrupts," are triggered by a software instruction and operate similarly to a jump or branch instruction. An external interrupt, or a "hardware interrupt," is caused by an external hardware module. As an example, many computer systems use **interrupt driven I/O**, a process, where pressing a key on the keyboard or clicking a button on the mouse triggers an interrupt. The processor stops what it is doing, it reads the input from the keyboard or mouse, and then it returns to the current program.

The image below shows conceptually how an interrupt happens:

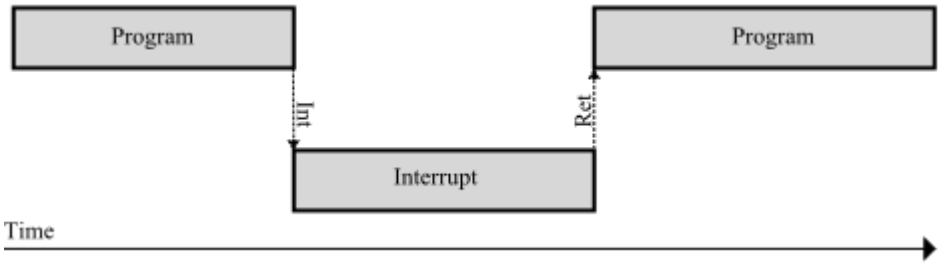


Figure 3. 8

The grey bars represent the control flow. The top line is the program that is currently running, and the bottom bar is the interrupt service routine (ISR). Notice that when the interrupt (**Int**) occurs, the program stops executing and the microcontroller begins to execute the ISR. Once the ISR is complete, the microcontroller returns to processing the program where it left off.

Interrupt performance depends on both hardware and software. The two most important properties of interrupt performance are the IRQ latency and throughput.

Modern hardware and software systems are often so complicated that it's practically impossible to deduce interrupt latency ahead of time by looking at source code and datasheets, so often it is directly measured by experiment.

Some processors have instructions that take many cycles to execute, making latency worse. Some people suggest avoiding such high-latency instructions.

Other processors are specifically designed to give predictable and lower latency response times.

Another method is polling where the micro controller continuously monitors the status of a device or a particular condition and waits until the required status or condition is met and then it performs the service, after performing this task it executes the next task. For example, just take a look at our section of UART code you can see that we are continuously monitoring the bit 5 of UART0_FR_R register to ensure that any previous transmission has been completed.

```
while((UART0_FR_R & 0x00000010) != 0) {
    ;
}
UART0_DR_R = data;
```

If yes, then transmit the next incoming data, while in interrupt the same thing is serviced by a microcontroller without continuously monitoring the status of a device though it serves when it gets notified by the device by receiving an interrupt signal.

ISR (Interrupt Service Routine)

For every interrupt there must be a program associated with it. When an interrupt occurs, this program is executed to perform certain service for the interrupt. This program is commonly referred to as an interrupt service routine (ISR) or interrupt handler. When an interrupt occurs, the CPU runs the interrupt service routine. Now the question is, how the ISR gets executed? As shown in the Figure 1, in the ARM CPU there are pins that are associated with hardware interrupts. They are input signals into the CPU. When the signals are triggered, CPU pushes the PC register onto the stack and loads the PC register with the address of the interrupt service routine. This causes the ISR to get executed.

Interrupt Vector Table:

Since there is a program (ISR) associated with every interrupt and this program resides in memory (RAM or ROM), there must be a look-up table to hold the addresses of these ISRs. This look-up table is called interrupt vector table. In the ARM, the lowest 1024 bytes (256 * 4 = 1024) of memory space are set aside for the interrupt vector table and must not be used for any other function. Of the 256 interrupts, some are used for software interrupts, and some are for hardware IRQ interrupts.

Interrupt #	Interrupt	Memory Location	Priority Level
0	Stack Pointer Initial Value	0x00000000	
1	Reset	0x00000004	-3 Highest
2	NMI	0x00000008	-2
3	Hard Fault	0x0000000C	-1
4	Memory Management Fault	0x00000010	Programmable
5	Bus Fault	0x00000014	Programmable
6	Usage Fault (undefined instructions, divide by zero,	0x00000018	Programmable

	unaligned memory access, ...)		
7	Reserved	0x0000001C	Programmable
8	Reserved	0x00000020	Programmable
9	Reserved	0x00000024	Programmable
10	Reserved	0x00000028	Programmable
11	SVCall	0x0000002C	Programmable
12	Debug Moniter	0x00000030	Programmable
13	Reserved	0x00000034	Programmable
14	PendSV	0x00000038	Programmable
15	SysTick	0x0000003C	Programmable
16	IRQ for peripherals	0x00000040	Programmable
17	IRQ for peripherals	0x00000044	Programmable
...	...	...	...
255	IRQ for peripherals	0x000003FC	Programmable

A detailed default vector table can be found at ***tm4c123gh6pm_startup_ccs_gcc.c*** file.

Nested Vectored Interrupt Controller (NVIC):

It is a control unit for a cortex-M4 MCU, it provides the group of programmable registers where all the exceptions and interrupts, including maskable and non-maskable interrupts are handled and preprocessed in a specific sequences.

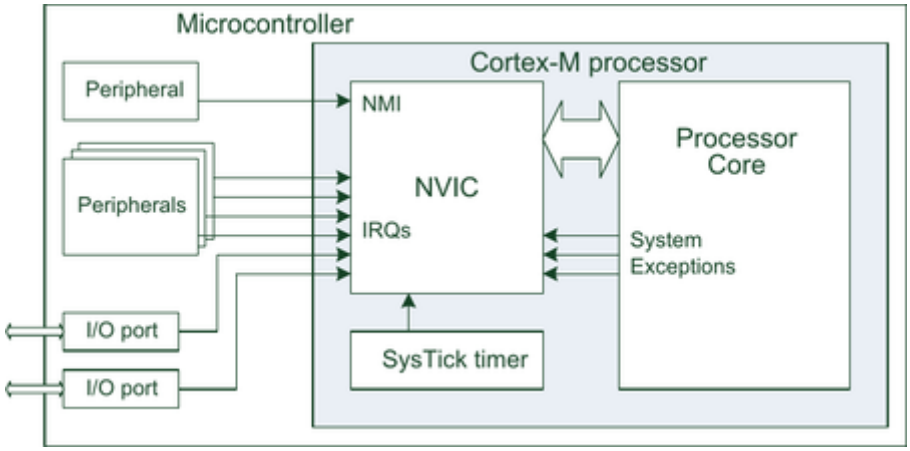


Figure 3. 9

Figure 1: NVIC in ARM Cortex-M

- Interrupts on the Cortex-M are controlled by the Nested Vectored Interrupt Controller (NVIC).
- Each exception has an associated 32-bit vector that points to the memory location where the ISR that handles the exception is located.

With CCS IDE, a new project will get a C startup code **tm4c123gh6pm_startup_ccs_gcc.c** created by the project wizard. For each interrupt, there is a dummy interrupt hander that does not perform any thing and will never return from the handler. The addresses of these interrupt handlers are listed in the interrupt vector table named *g_pfnVectors* in the file. You need to carefully find the appropriate vector position and replace *IntDefaultHandler* with the name of your Interrupt handler. The linker will overwrite the interrupt vector table with the new interrupt handler. The interrupt handler is written with a format of a function in C language.

Interrupt Priority for ARM Cortex-M

- All exceptions and interrupts in the Cortex-M4 system have certain priority levels, either maskable or unmaskable sources.
- Most maskable interrupts have programmable priority levels, but all Non-Maskable Interrupts (NMIs) have fixed priority levels.

- When an exception or interrupt occurs, the NVIC performs a comparison between the priority level of current exception or interrupt and the priority level of the new coming exception/interrupt. The current running task will be suspended, and the control will be transferred to the service routine of the new coming exception/interrupt if the priority level of the new coming exception/interrupt is higher.
- In the ARM Cortex-M4 system, the interrupt priority levels are controlled by the Interrupt Priority Registers, as shown in Table 7.3.
- Each priority register can use 3 bits, 4 bits, or 8 bits to cover all priority levels used in the priority control system.
- A total of 8 priority levels can be used if 3 bits are used in this register, and 16 priority levels can be obtained if 4 bits are used in this register.
- Devices within the Tiva family support up to 154 interrupt sources and 8 priority levels, which means that 3 bits are used in the priority register in the TM4C123GH6PM MCU.
- To activate an interrupt source, we need to set its priority and enable that source in the NVIC. This activation is in addition to the arm and enable steps.

To arm a device means to allow the hardware trigger to interrupt. Conversely, to disarm a device means to shut off or disconnect the hardware trigger from the interrupts

The following five conditions must be true for an interrupt to be generated:

1. Device arm

Each potential interrupt trigger has a separate **arm** bit that the software can activate or deactivate. The software will set the arm bits for those devices from which it wishes to accept interrupts and will deactivate the arm bits within those devices from which interrupts are not to be allowed. In other words it uses the arm bits to individually select which devices will and which devices will not request interrupts.

2. NVIC enable

For most devices there is a enable bit in the NVIC that must be set (periodic SysTick interrupts are an exception, having no NVIC enable).

3. Global enable

Bit 0 of the special register **PRIMASK** is the interrupt mask bit, **I**. If this bit is 1 most interrupts and exceptions are not allowed, which we will define as disabled. If the bit is 0, then interrupts are allowed, which we will define as enabled.

4. Interrupt priority level must be higher than current level executing

The **BASEPRI** register prevents interrupts with lower priority interrupts, but allows higher priority interrupts. For example if the software sets the **BASEPRI** to 3, then requests with level 0, 1, and 2 can interrupt, while requests at levels 3 and higher will be postponed. The software can also specify the priority level of each interrupt request. If **BASEPRI** is zero, then the priority feature is disabled, and all interrupts are allowed.

5. Hardware event trigger.

Hardware triggers are bits in the GPIO_PORTx_RIS_R register that are set on rising or falling edges of digital input pins.

For an interrupt to occur, these five conditions must be simultaneously true but can occur in any order.

Interrupts in our project

Unless you are making a very simple software, you will most always use interrupts in your Embedded software. Our project is no different, as we use interrupts in some of modules.

Timers are one of the examples, as whenever a timer finishes, it interrupts the hardware to handle a certain block of code.

UART too, uses interrupts, so that when a microcontroller sends some information to another, it interrupts the latter to handle the sent information.

GSM uses interrupt as well, so that when remote data is sent to the hardware system, it interrupts the system to deal with the sent data.

3.2 Hardware Background:

In this section we will cover the used hardware components in detail and describe how they work and use of some of them in our system

Table of used Hardware Components:

Hardware	Hardware
Tiva C Boards (TM4C123GH6PM)	RFID Module (RC522)
Raspberry Pi 4 (Model B)	Bread Boards
Raspberry Pi Camera Module 2 Noir	Jumper Wires
GPS Module (SIM808)	EEPROM (24C16)
GSM Module (SIM808)	LCD (4x20)
Power Adapters	KEYPAD (4x4)

Tiva C (TM4C123GH6PM):

The Tiva™ C Series TM4C123G LaunchPad Evaluation Board (EK-TM4C123GXL) is a low-cost evaluation platform for ARM® Cortex™-M4F-based microcontrollers. The Tiva C Series LaunchPad design highlights the TM4C123GH6PMI microcontroller USB 2.0 device interface, hibernation module, and motion control pulse-width modulator (MC PWM) module. The Tiva C Series LaunchPad also features programmable user buttons and an RGB LED for custom applications. The stackable headers of the Tiva C Series TM4C123G LaunchPad BoosterPack XL interface demonstrates how easy it is to expand the functionality of the Tiva C Series LaunchPad when interfacing to other peripherals on many existing BoosterPack add-on boards as well as future products. Figure 1-1 shows a photo of the Tiva C Series LaunchPad.

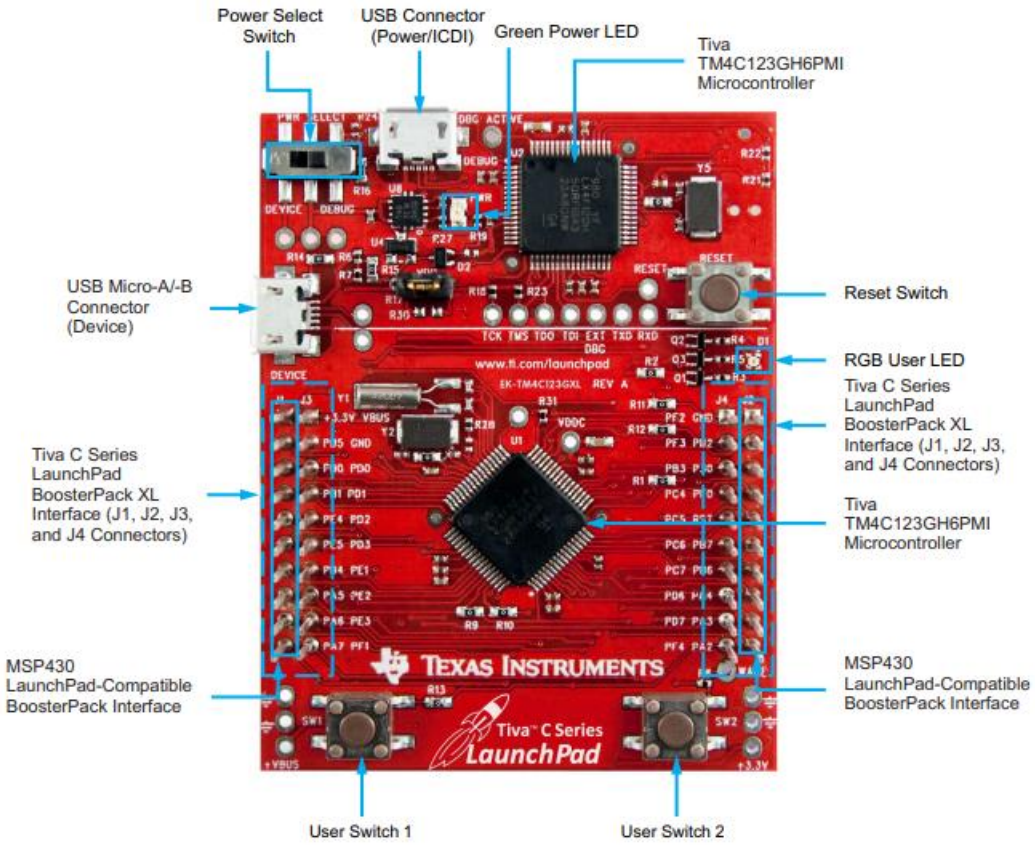


Figure 3. 10

Features:

Your Tiva C Series LaunchPad includes the following features:

- Tiva TM4C123GH6PMI microcontroller • Motion control PWM
- USB micro-A and micro-B connector for USB device, host, and on-the-go (OTG) connectivity • RGB user LED
- Two user switches (application/wake)
- Available I/O brought out to headers on a 0.1-in (2.54-mm) grid
- On-board ICDI • Switch-selectable power sources: – ICDI – USB device
- Reset switch

- Preloaded RGB quickstart application
- Supported by TivaWare for C Series software including the USB library and the peripheral driver library • Tiva C Series TM4C123G LaunchPad BoosterPack XL Interface, which features stackable headers to expand the capabilities of the Tiva C Series LaunchPad development platform.

Tiva™ C Series Overview:

The Tiva™ C Series ARM Cortex-M4 microcontrollers provide top performance and advanced integration. The product family is positioned for cost-conscious applications requiring significant control processing and connectivity capabilities such as:

- Low power, hand-held smart devices
- Gaming equipment
- Home and commercial site monitoring and control
- Motion control
- Medical instrumentation
- Test and measurement equipment
- Factory automation
- Fire and security
- Smart Energy/Smart Grid solutions
- Intelligent lighting control
- Transportation

Used Peripherals:

- UART
- TIMERS
- I2C
- SPI

- NVIC

Our Project:

In our project we used 2 Tiva C LaunchPads as our microcontrollers to control the various activities of the application

Control ECU Block:

This Tiva C is responsible for controlling the Authentication process and making sure that the user Passcode, RFID tag and Face ID matches the ones saved on the system, it also communicates with the other Tiva C (HMI ECU) and the Raspberry pi via UART protocol

HMI ECU Block:

LCD, keypad and RFID are all connected to this Tiva C which acts as the human interface part of the system where the user will interact with mentioned components.

TM4C123GH6PM Pinout:

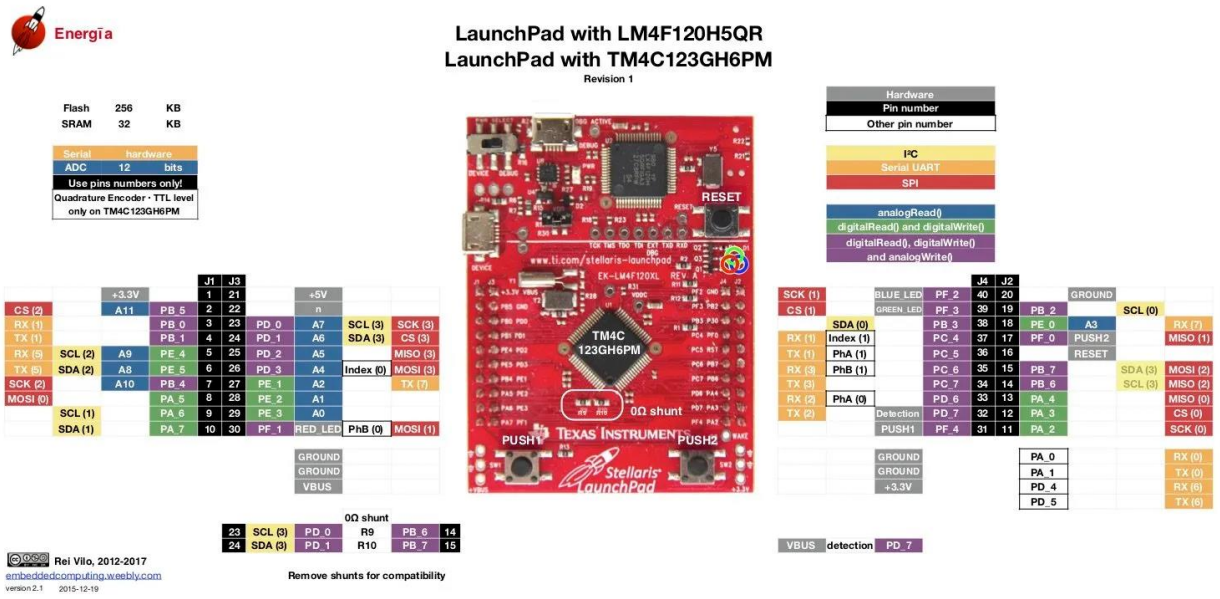


Figure 3. 11

TM4C123GH6PM Microcontrollers UART Ports:

As we mentioned earlier, Tiva C launchpad has TM4C123GH6PM Microcontroller. This TI microcontroller supports 8 UART ports, starting from UART0 to UART7. On top of that each module has a separate 4 bytes transmit and receive FIFO storage that reduces the CPU interrupt service load. The following table provides the pinout of each UART port:

UART Module	Rx pin	Tx pin
UART0	PA0	PA1
UART1	PC4	PC5
UART2	PD6	PD7
UART3	PC6	PC7
UART4	PC4	PC5
UART5	PE4	PE5

UART6	PD4	PD5
UART7	PE0	PE1

Raspberry Pi 4 (Model B 8GB RAM):

Overview:

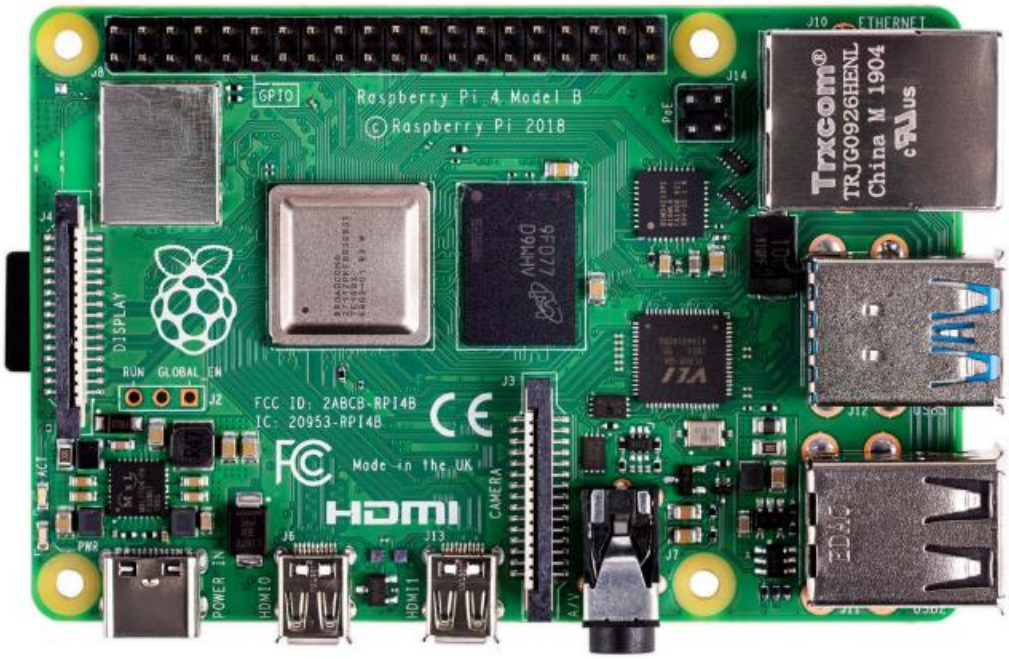


Figure 3. 12

Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI

ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

Specification:

Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory	2GB, 4GB or 8GB LPDDR4 (depending on model)
Connectivity	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & Sound	2 × micro-HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port

	2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimedia	H.265 (4Kp60 decode) H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD card support	Micro SD card slot for loading operating system and data storage
Input power	5V DC via USB-C connector (minimum 3A1) 5V DC via GPIO header (minimum 3A1) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment	Operating temperature 0–50ºC
Production lifetime	The Raspberry Pi 4 Model B will remain in production until at least January 2026.

Software & OS:

Raspberry Pi OS (previously called Raspbian) is the recommended operating system for normal use on a Raspberry Pi.

Tech Specs:

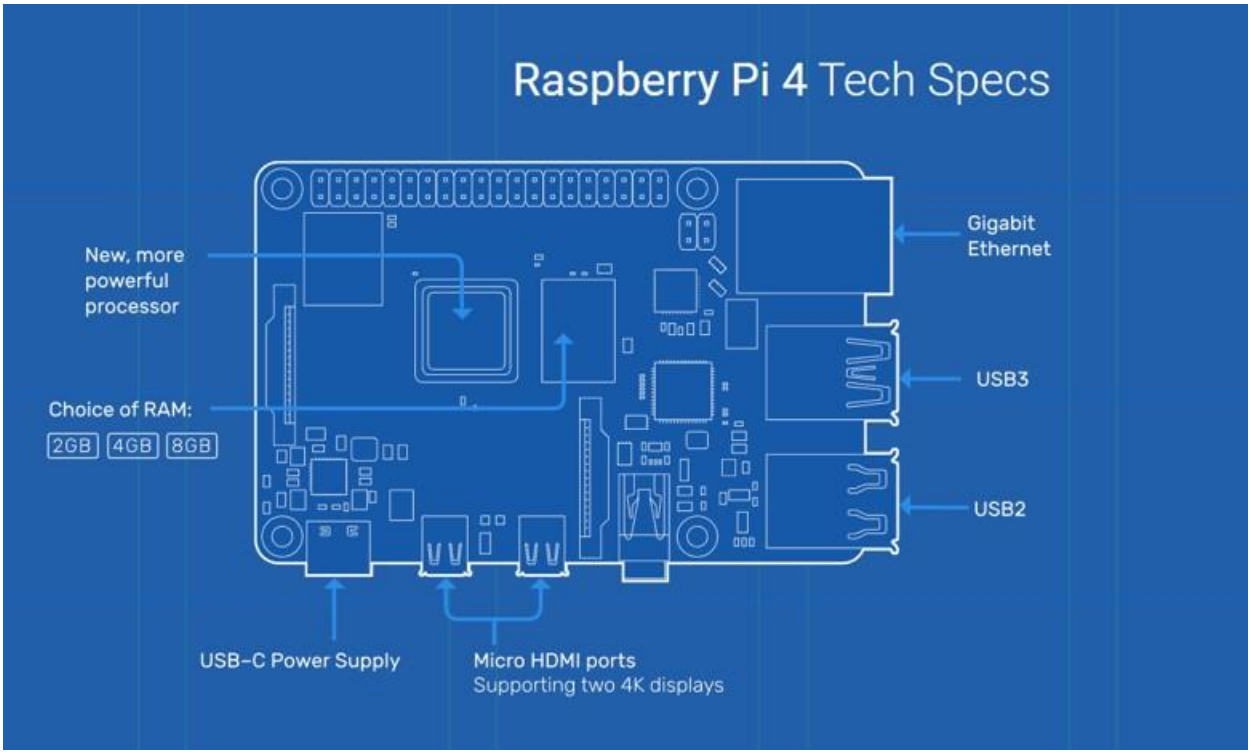


Figure 3. 13

Raspberry Pi 4 Pinout:

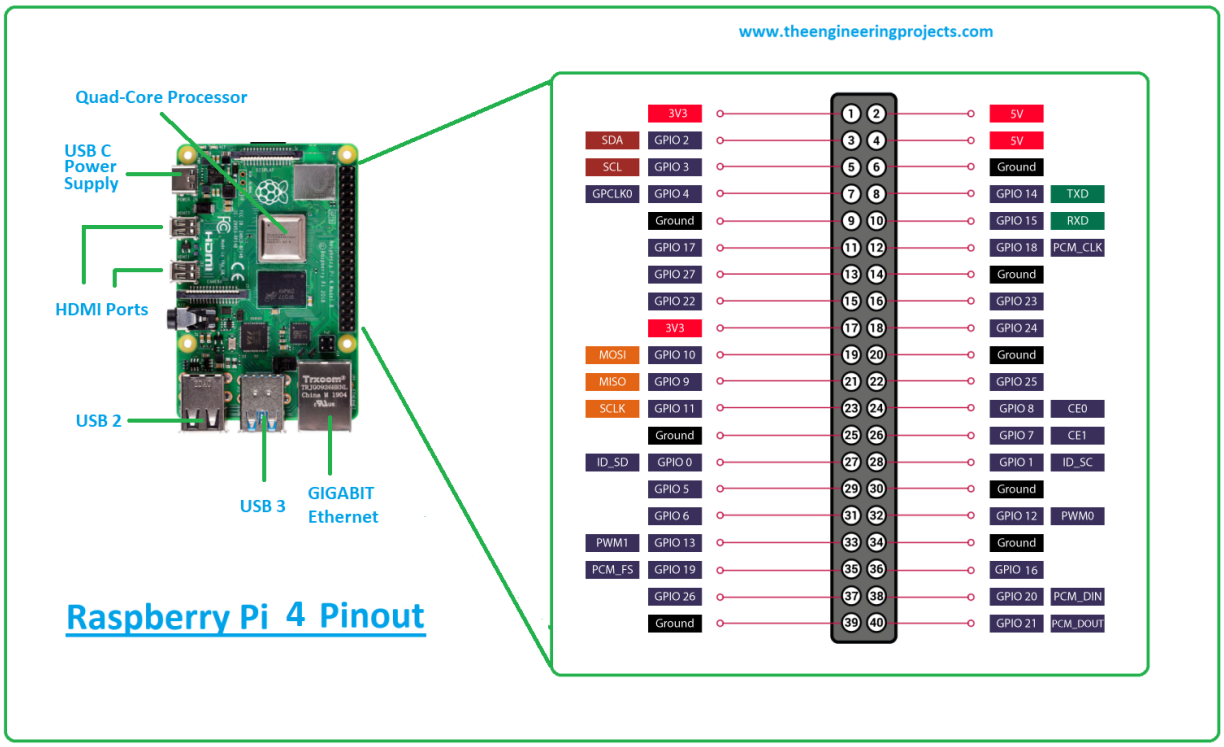


Figure 3. 14

Power and Ground:

This board comes with three types of power pins.

1. 5V
2. 3V3 (3.3V)
3. Ground (0V)

For example, if you have PIR or humidity sensors, you can use these power pins to power up those sensors.

GPIO Pins:

- GPIO pins are general-purpose input/output pins that are used for connection with external devices. These pins can be configured to either general-purpose input or general-purpose output pins or as one of up to six special settings those functions are pin-dependent.
- External labels (from GPIO2 to GPIO27) come with the Broadcom (BCM) naming convention. This convention is useful when you are going to program with Python libraries.
- Internal labels (from 1 to 40) project the Board naming convention. This convention is useful when BCM is not supported. It is used with some programming libraries.

UART Pins:

- This board also features UART serial communication protocol. The UART serial port comes with two pins Rx and Tx.
- The Tx is the transmission pin that is used for the transmission of serial data and Rx is the receiving pin that guarantees the receiving of serial data.

Ways to program the Raspberry PI 4 Board:

You can control the Raspberry Pi 4 GPIO pins using many programming languages. Some of the popular languages along with learning material is given below:

- GPIO Programming using **Python**
- Programming GPIO with **C/C++** using standard **kernel interface via libgpiod**

Raspberry Pi Camera Module 2 Noir:

Figure 3. 15

Features:

- 8-megapixel camera capable of taking infrared photographs of 3280 x 2464 pixels
- Capture video at 1080p30, 720p60 and 640x480p90 resolutions
- All software is supported within the latest version of Raspbian Operating System.

Overview:

The Raspberry Pi NoIR Camera Module v2 is a high quality 8-megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464-pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSi interface, designed especially for interfacing to cameras.

The board itself is tiny, at around 25mm x 23mm x 9mm. It also weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. It connects to Raspberry Pi by way of a short ribbon cable. The high-quality Sony sensor itself has a native resolution of 8 megapixel and has a fixed focus lens on-board. In terms of still images, the camera is capable of 3280 x 2464-pixel static images, and also supports 1080p30, 720p60 and 640x480p90 video. The NoIR Camera has No InfraRed (NoIR) filter on the lens which makes it perfect for doing Infrared photography and taking pictures in low light (twilight) environments.

Applications:

- Infrared photography
- Low light photography
- Monitoring plant growth
- CCTV security camera

Project Functionality:

The main purpose for the existence of the raspberry pi in our system is that to connect the raspberry pi camera to it and process the image taken by it using the face recognition algorithm and based on the output of the algorithm the raspberry pi shall act and send a certain bit via UART Protocol to the Control Block.

RC522 RFID Reader/Writer Module:

Overview:

The RC522 RFID module based on **MFRC522 IC from NXP** is one of the most inexpensive RFID options that you can get. It usually comes with a RFID card tag and key fob tag having **1KB memory**. And best of all, it can write a tag, so you can store some sort of secret message in it.

The RC522 RFID Reader module is designed to create a **13.56MHz** electromagnetic field that it uses to communicate with the RFID tags (ISO 14443A standard tags). The reader can communicate with a microcontroller over a 4-pin **Serial Peripheral Interface (SPI)** with a maximum data rate of **10Mbps.** It also supports communication over **I2C** and **UART** protocols, in our project **UART** communication protocol is used instead of **SPI**.

The module comes with an interrupt pin. It is handy because instead of constantly asking the RFID module “is there a card in view yet? “, the module will alert us when a tag comes into its vicinity.

The operating voltage of the module is from **2.5 to 3.3V**, but the good news is that the **logic pins are 5-volt tolerant**, so we can easily connect it to the **Tiva C** board or any 5V logic microcontroller without using any logic level converter.

For our project RFID is **HAL** Software Driver is divided into (RFID.h & RFID.c) files containing the module registers, commands and most importantly module functions, the module should be connected to the HMI block which mean the user will interact with it

RFID Module Structure and Pinout:

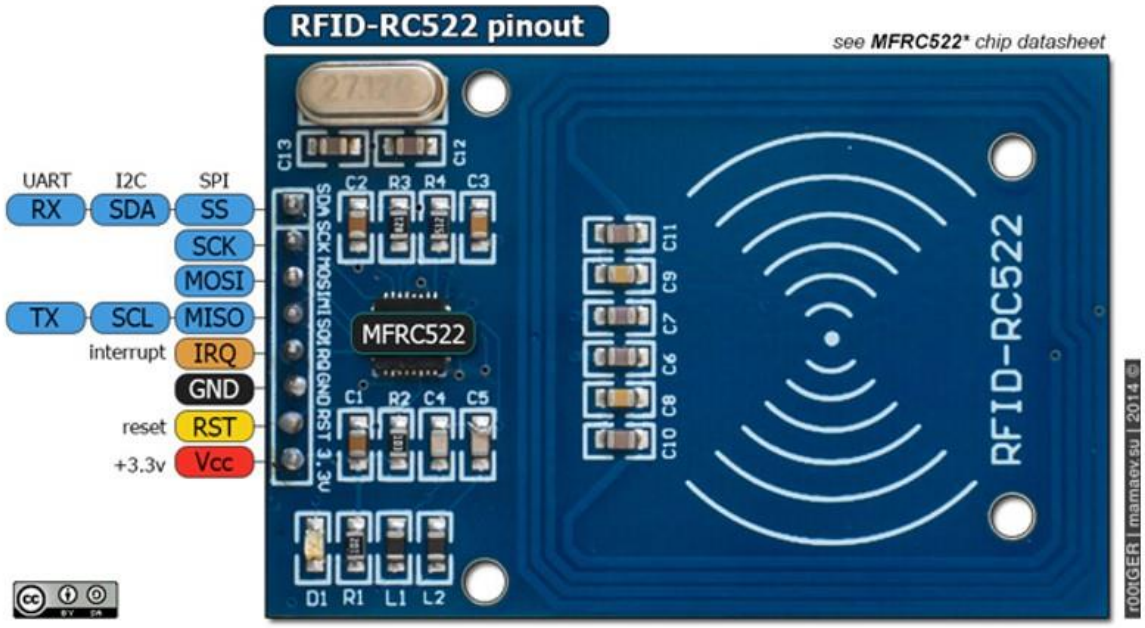


Figure 3. 16

The Module consists of **MFRC522 which** is a highly integrated reader/writer IC for contactless communication at 13.56 MHz

The MFRC522's internal transmitter part is able to drive a reader/writer antenna designed to communicate with **ISO/IEC 14443A/MIFARE®** cards and transponders without additional active circuitry. The receiver part provides a robust and efficient implementation of a demodulation and decoding circuitry for signals from **ISO/IEC 14443A/MIFARE®** compatible cards and transponders.

The RC522 module has total 8 pins that interface it to the outside world. The connections are as follows:

Pin Number	Pin Name	Description
1	Vcc	Used to Power the module, typically 3.3V is used
2	RST	Reset pin – used to reset or power down the module
3	Ground	Connected to Ground of system
4	IRQ	Interrupt pin – used to wake up the module when a device comes into range
5	MISO/SCL/Tx	MISO pin when used for SPI communication, acts as SCL for I2c and Tx for UART.
6	MOSI	Master out slave in pin for SPI communication
7	SCK	Serial Clock pin – used to provide clock source
8	SS/SDA/Rx	Acts as Serial input (SS) for SPI communication, SDA for IC and Rx during UART

RC522 Features:

- 13.56MHz RFID module
- MFRC522 chip-based board
- Operating voltage: 2.5V to 3.3V
- Communication: SPI, I2C protocol, UART
- Maximum Data Rate: 10Mbps
- Read Range: 5cm
- Current Consumption: 13-26mA
- Power down mode consumption: 10uA (min)

How to use RC522 RFID Module:

The RC522 has an operating voltage between 2.5V to 3.3V and hence is normally powered by 3.3V and should be used with 3.3V communication lines. But the communication pins of this module are 5V tolerant and hence it can be used with 5V microcontrollers also like Arduino without any additional hardware. The module supports SPI, IIC and UART communication but out of these SPI is often used since it is the fasted with a maximum data rate of 10Mbps.

Since in application, most of the time reader module will be waiting for the tag to come into proximity. The Reader can be put into power down mode to save power in battery operated applications. This can be achieved by using the IRQ pin on the module. The minimum current consumed by the module during power down mode will be 10uA only.

Applications:

- Automatic billing systems
- Attendance systems
- Verification/Identification system
- Access control systems

Project Functionality:

In the implementation of our project The RC522 RFID Module will be used as one of the three main authentication methods along with the keypad and face id to identify the owner, the user will have to use the RFID tag in order to activate the fuel pump via the security system.

4X4 Keypad:

Overview:

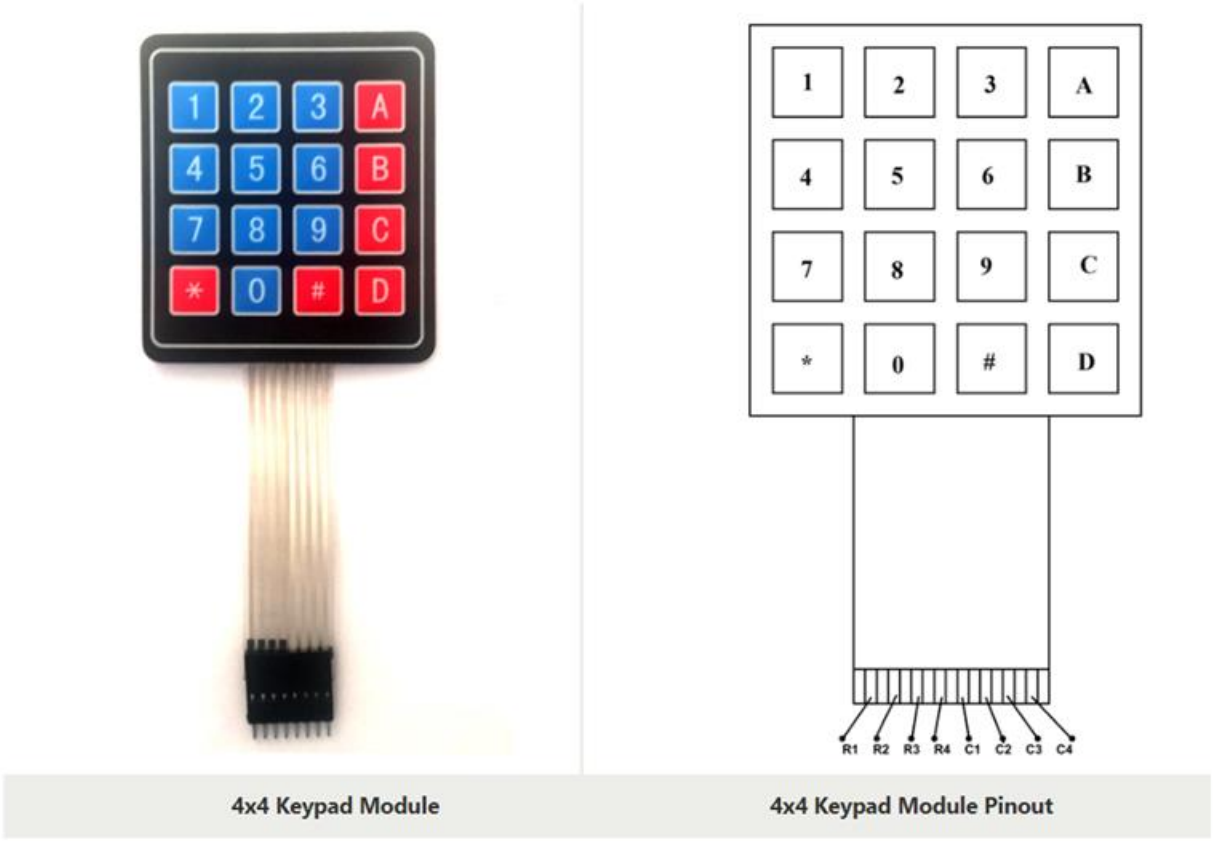


Figure 3. 17

4X4 KEYPAD Pin Configuration

4X4 KEYPAD MODULES are available in different sizes and shapes. But they all have same pin configuration. It is easy to make 4X4 KEYPAD by arranging 16 buttons in matrix formation by yourself.

Pin Number	Description
ROWS	
1	PIN1 is taken out from 1st ROW
2	PIN2 is taken out from 2nd ROW
3	PIN3 is taken out from 3rd ROW
4	PIN4 is taken out from 4th ROW
COLUMN	
5	PIN5 is taken out from 1st COLUMN
6	PIN6 is taken out from 2nd COLUMN
7	PIN7 is taken out from 3rd COLUMN
8	PIN8 is taken out from 4th COLUMN

As given in above table, a **4X4 KEYPAD** will have **EIGHT TERMINALS**. In them four are **ROWS of MATRIX** and four are **COLUMNS of MATRIX**.

These 8 PINS are driven out from 16 buttons present in the MODULE. Those 16 alphanumeric digits on the MODULE surface are the 16 buttons arranged in MATRIX formation.

The **internal structure of 4X4 KEYPAD MODULE** is shown below.

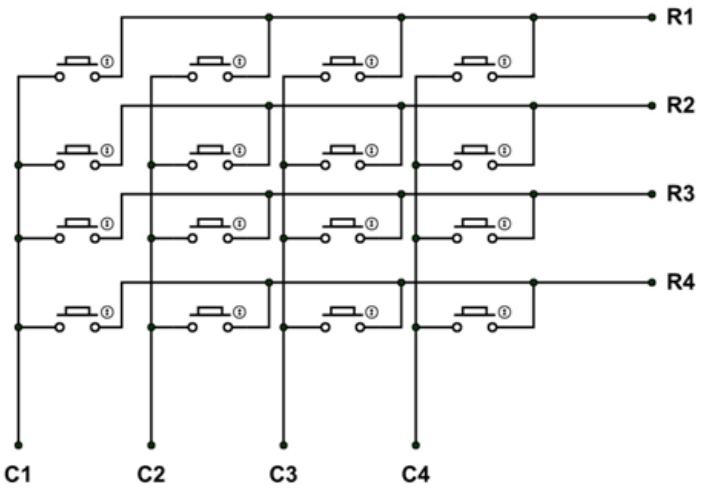


Figure 3. 18

4X4 KEYPAD MODULE Features and Specifications

- Maximum Voltage across EACH SEGMENT or BUTTON: 24V
- Maximum Current through EACH SEGMENT or BUTTON: 30mA
- Maximum operating temperature: 0°C to + 50°C
- Ultra-thin design
- Adhesive backing
- Easy interface
- Long life.

Note: Complete technical information can be found in the **4x4 Keypad Module Datasheet** linked at the bottom of this page.

Where 4X4 KEYPAD MODULE is Used?

4X4 KEYPAD MODULE is used for many kinds of applications. Few are:

Case1: When easy to use INPUT device is needed. For control systems or embedded systems there will be a time where human interaction is needed. Like a coffee machine needs choice selection. In those cases, using 4X4 KEYPAD MODULE is best solution.

Case2: Input module can only have few pins. If certain applications are developed of processors with few pins, then that system cannot afford to provide more pins to INPUT DEVICE. At that time using KEYPAD MODULE is ideal, as it can give 16 key inputs for using 8 terminals.

Case3: For professional look. Using KEYPAD MODULE can give more professional view to the system than other key INPUT DEVICES

Applications

- Security systems.
- Vending machines.
- Industrial machines.
- Engineering systems.
- Measuring instruments.
- Data entry for Embedded Systems
- Hobby projects.
- Basically, anywhere INPUT device is needed.

Liquid Crystal Display (LCD):

The LCD stands for liquid crystal display that works on the light modulation features of liquid crystals. It is available in electronic visible display, video display and flat panel display. There are numerous categories and features that exist in markets of LCD, and you can see it on your mobile, laptop, computer, and

television screen. The invention of LCD gives new life to electronic industries and replaces led and gas plasma techniques. It also replaces the CTR (cathode ray) tube that used for visual display. The input power consumed by the liquid crystal display is less then light-emitting diode and plasma display.

(20x4) LCD module:

1. In a 20×4 LCD module, there are four rows in display and in one row twenty character can be displayed and in one display eighty characters can be shown.
2. This liquid crystal module uses HDD44780 (It is a controller used to display monochrome text displays) parallel interfacing.
3. The liquid crystal display interfacing code is easily accessible. We just require eleven input and output for the interfacing of the LCD screen.
4. The input supply for this module is three volts or five volts; with that module other components like PIC, Raspberry pi, Tiva c.
5. This electronic device can be used in different embedded systems, industries, medical devices, and portable devices like mobile, watches, laptops.
6. Liquid crystal display works on two types of the signal first one is data and the second one is for control.
7. The existence of these signals can be identified through the on and off condition of RS pin. Data can be read by pushing the Read/write pin.

Features of LCD (20 X 4):

1. The cursor of this module has 5×8 (40) dots.
2. On this module already assembled the controller of RW1063.
3. This module operates on five volts input supply and can also work on three volts.
4. The three volts can also be used for the negative supply.

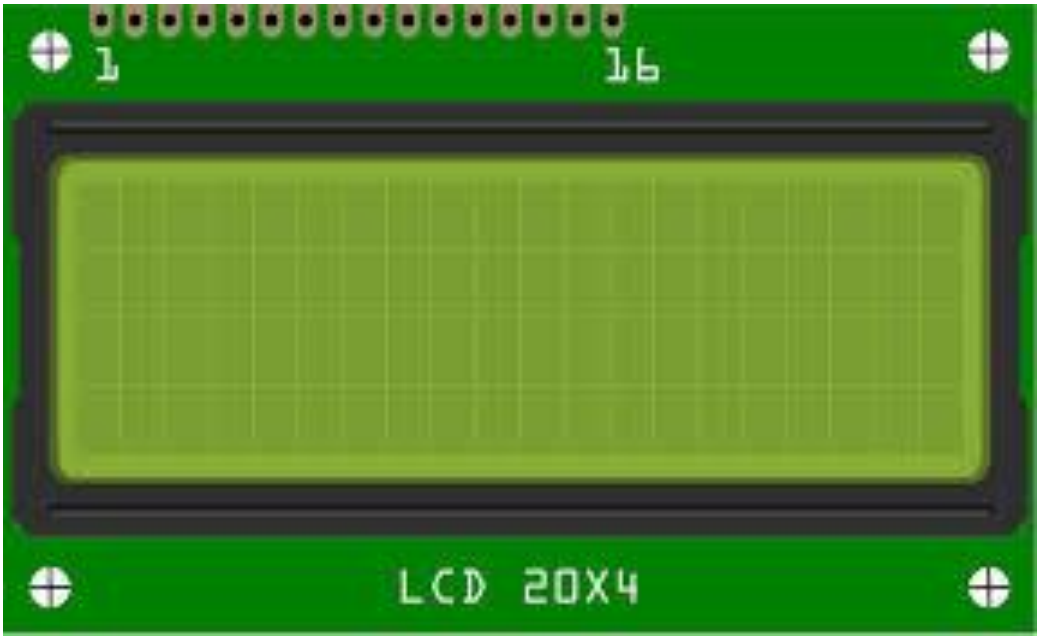


Figure 3. 19

Functions of pins in LCD (20 x 4):

Pin No:	Pin Name:	Parameters
Pin#1	It denoted as Vss	It is ground pinout potential at this pinout is zero.
Pin#2	It denoted as Vdd	At this pinout, five volts are provided.
Pin#3	This pinout denoted as Vo	This pinout is used to set the contrast of the screen.
Pin#4	This pin denoted as RS	It used to H/L register select signal.
Pin#5	It denoted as R/W	It used for H/L read/write signal.
Pin#6	This pinout denoted as E	It used for H/L enable signal.
Pin#7-14	The pinouts from seven to fourteen denoted as DB0 – DB7.	It used for H/L data bus for 4 bit or 8-bit mode.
Pin#15	It identified as A (LED+)	It used to set backlight anode.
Pin#16	It recognized as K (LED-).	It used to set backlight cathode.

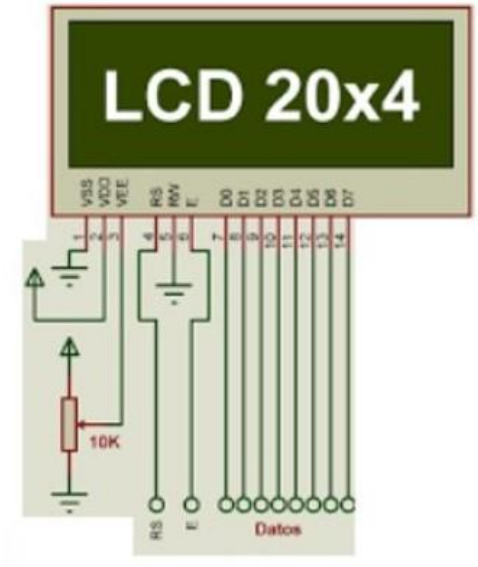


Figure 3. 20

Advantages of LCD:

1. It is less expensive, lightweight as compared to the cathode ray tube display.
2. It uses less power according to the brightness resolution.
3. It produces less amount of heat due to less use of power.
4. In this module, there is no geometric distortion.
5. It can be constructed in any shape and size according to user requirements.
6. The LCD used in the computer monitor uses twelve volts.
7. Extremely high resolution.
8. It is not affected by magnetic fields.

Disadvantages of LCD:

1. It requires an additional light source.
2. Limited viewing angle and brightness.
3. In high-temperature environments, there is a loss of contrast.
4. It consumes a lot of electricity therefore a lot of power is consumed.

GPS/GSM Module (SIM808):

GPS Module:

In view of automobile anti-theft systems mostly used static real-time detection and alarm at present, in our system we design an automobile anti-theft system based on GSM and GPS modules. The system is developed to detect automobile stolen to the automobile owner. Automobile location can be obtained with the GPS module integrated in anti-theft system. The system can keep in touch with automobile owner through the GSM module, to monitor the safety and reliability of automobile. By far, the best way to secretly track a car is to install a Low Star device. Powered by GPS technology, Low Star links to an app on your smartphone, which allows you to accurately pinpoint the exact location of your vehicle in real time, even while it's still moving Basically, the SIM808 module is designed for the global market. It is integrated with a high-performance GSM/GPRS chip, and as a bonus, it also has a GPS engine and a BT engine. SIM808 is a quad-band GSM/GPRS module that works on frequencies GSM 850MHz, EGSM 900MHz, DCS 1800MHz, and PCS 1900MHz. It features GPRS multi-slot class 12/class 10 and supports the GPRS coding schemes CS-1, CS-2, CS-3, and CS-4. The GPS solution offers best-in-class acquisition and tracking sensitivity, time-to-first fix (TTFF), and accuracy.

Benefits of the SIM808 Module:

The SIM808 has 68 pins SMT pads (a single chip module which comes without soldering), which provides all hardware interfaces between the module and the user's boards. It is implemented in this project with our proposed system that also communicate with the following features:

1. One full modem serial port (UART interface).
2. One SIM card interface.
3. Ability to support 4*4keypads by default.
4. A charging interface one USB, which supports debug and firmware.
5. Ability to support GPS function.

6. 8. A power-saving technique the keeps the current consumption is as low as 1.2mA in sleep mode (with GPS engine powered down) and PCM / SPI interface.

GPS determines the distance between a GPS satellite and a GPS receiver by measuring the amount of time taken by a radio signal (the GPS signal) to travel from the satellite to the receiver. To obtain accurate information, the satellites and the receiver use very accurate clocks, which are synchronized so that they generate the same code at exactly the same time.
This GPS based vehicle tracking system implements RS-232 protocol for serial communication between the microcontroller, GPS and GSM modem.

Software Program:

The program for the microcontroller is written in 'C' language and compiled using *IAR Embedded Workbench for* Arm. Complete development environment for Arm, generating fast, compact code and enabling you to take full control of your code.

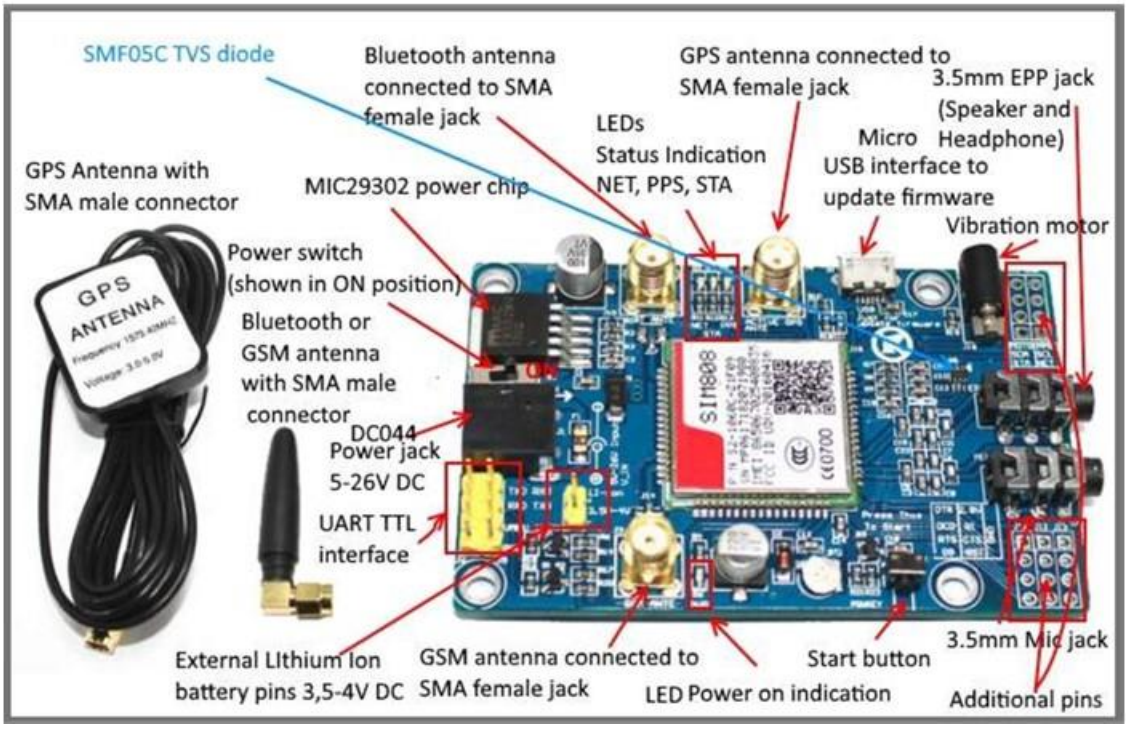


Figure 3. 21

SIM 808 key features:

Feature	Implementation
Power supply	3.4V ~ 4.4V
Power saving	Typical power consumption in sleep mode is 1mA (BS-PA-MFRMS=9, GPS engine is powered down)
Charging	Supports charging control for Li-Ion battery

Serial port and USB interface	**Serial port:** ● Full modem interface with status and control lines, unbalanced, asynchronous. ● 1200bps to 115200bps. ● Can be used for AT commands or data stream. ● Support RTS/CTS hardware handshake and software ON/OFF flow control. ● Multiplex ability according to GSM 07.10 Multiplexer Protocol. ● Autobauding supports baud rate from 1200 bps to 115200bps. **USB interface:** ● Can be used as debugging and firmware upgrading.
Transmitting power	● Class 4 (2W) at GSM 850 and EGSM 900 ● Class 1 (1W) at DCS 1800 and PCS 1900
GPRS connectivity	● GPRS multi-slot class 12（default） ● GPRS multi-slot class 1~12 (optional)
Temperature range	● Normal operation: -40℃ ~ +85℃ ● Storage temperature -45℃~ +90℃
Data GPRS	● GPRS data downlink transfer: max. 85.6 kbps ● GPRS data uplink transfer: max. 85.6 kbps ● Coding scheme: CS-1, CS-2, CS-3 and CS-4 ● PAP protocol for PPP connect ● Integrate the TCP/IP protocol. ● Support Packet Broadcast Control Channel (PBCCH) ● CSD transmission rates：2.4，4.8，9.6，14.4 kbps
CSD	● Support CSD transmission

SIM 808 functional diagram:

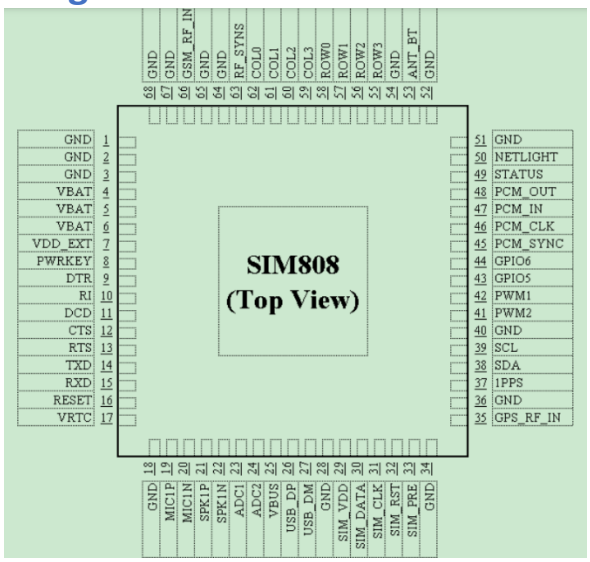


Figure 3. 22

GPS application interface:

We use AT commands for GPS function such as:

Command	Description
AT+CGPSPWR	GPS power control
AT+CGPSRST	GPS mode reset (hot/warm/cold)
AT+CGPSSTATUS	Get current GPS status
AT+CGPSOUT	GPS NMEA data output control
AT+CGPSINF	Get current GPS location info
AT+CGPSIPR	Set GPS NMEA output UART bps

GSM:

At present, the rate of crime is increasing rapidly because it is a kind of evidence from the fact that thefts became a matter of routine. Particularly these vehicles may incur huge losses on the part of the amount invested in these vehicles. To overcome this problem, there are numerous technologies are available in the market such as GPS, GSM and GPRS systems. In the present day, most of the vehicles are designed with GSM-based vehicle theft control systems, which protects from thefts even if they are parked in the parking area.

GSM (Global System for Mobile communication) is a digital mobile network that is widely used by mobile phone users all over the world. It is a digital cellular technology used for transmitting mobile voice and data services. The concept of GSM emerged from a cell-based mobile radio system at Bell Laboratories in the early 1970s.

GSM uses a variation of time division multiple access (TDMA) and is the most widely used of the three digital wireless telephony technologies: TDMA, GSM and code-division multiple access (CDMA). GSM digitizes and compresses data, then sends it down a channel with two other streams of user data, each in its own time slot. It operates at either the 900 megahertz (MHz) or 1,800 MHz frequency band.

GSM, together with other technologies, is part of the evolution of wireless mobile telecommunications that includes High-Speed Circuit-Switched Data (HSCSD), General Packet Radio Service (GPRS), Enhanced Data GSM Environment (EDGE) and Universal Mobile Telecommunications Service (UMTS).

Power supply:

The power supply range of SIM 808 is from 3.4 volt to 4 volt.

Why GSM?

Listed below are the features of GSM that account for its popularity and wide acceptance.

- Improved spectrum efficiency.
- International roaming.
- Low-cost mobile sets and base stations (BSs).
- High-quality speech.
- Compatibility with Integrated Services Digital Network (ISDN). and other telephone company services.
- Support for new services.

GSM in our antitheft systems:

In our project we use GPS modem in SIM 808 through which the circuit consists of a GSM and GPS based vehicle tracking system. It consists of a microcontroller, GPS module, GSM modem and a power supply. GPS module gets the location information from satellites in the form of latitude and longitude. The microcontroller processes this information and sends it to the GSM modem. The GSM modem then sends the information to the owner's mobile phone.

SMS using SIM 808:

SIM808 with Leonardo mainboard is the latest Arduino card developed for multi-purpose by DF Robot. The SIM808 module onboard is an integrated quad-band

GSM/GPRS and GPS satellite navigation technology module. 4-layer PCB design makes the powerful card into an UNO size, this can save time and cost for you.

This SIM808 we chosen is the latest version, compared with the existing SIM808 module on the market, the new module has a better stability. However, you may have to pay attention that the AT command of the GPS is not compatible with the old version of the SIM808 module.

Applications:

GPRS network nodes, remote communications, GPS positioning and monitoring in real time, the Internet of things (IoT).

Specifications of SIM808:

- Operating voltage: 5V
- Input Voltage: 5V(USB)
- Digital I/O pins: 20 (7 PWM: 3,5,6,9,10,11,13)
- Analog Input pins: 12
- Clock Speed ： 16 MHz
- Dimension: 73*54mm

SIM808 Figure:

Figure 3. 23

Reading SMS:

1. Send a message to the SIM card on the module, for example **test!**
2. The serial monitor will print a message **+SMTI: "SM",1,** and **1** shows the number of the message
3. Send **AT+CMGR=1** to read the message of 1. The message "Test!" should display

Making Voice Calls:

1. Open the serial monitor, you will get a ready notice.
2. Send the number for example **1009***9217**. If the number is incorrect, you will receive a warning.
3. If the call is answered, the monitor will print text like lable 3
4. If the call is rejected, the monitor will print text like lable 4
    - You can send a **Newline** character to hang up, i.e.
        a. **\n** in code.

b. **Do NOT type anything** and click **Send**/ press **Enter** on your keyboard in serial monitor.

Receive a Voice Call:

1. **Make a call** to the SIM card on the module after you got the message **Waiting for a call.**
2. The module would answer the coming call automatically, and you will get the message.
3. You can send a Newline to hang up.

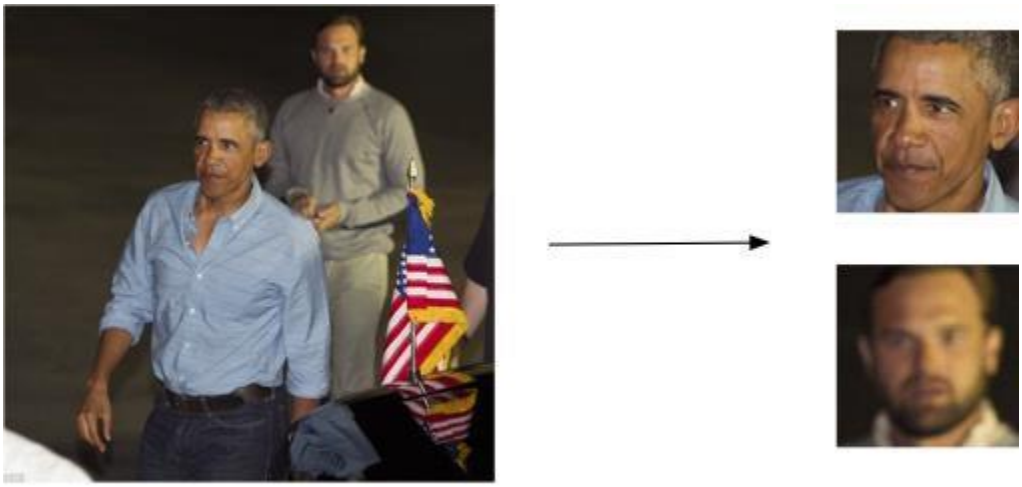
3.3 Face Recognition Algorithm:

Introduction:

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple `face_recognition` command line tool that lets you do face recognition on a folder of images from the command line!

Features

Find faces in pictures Find all the faces that appear in a picture:

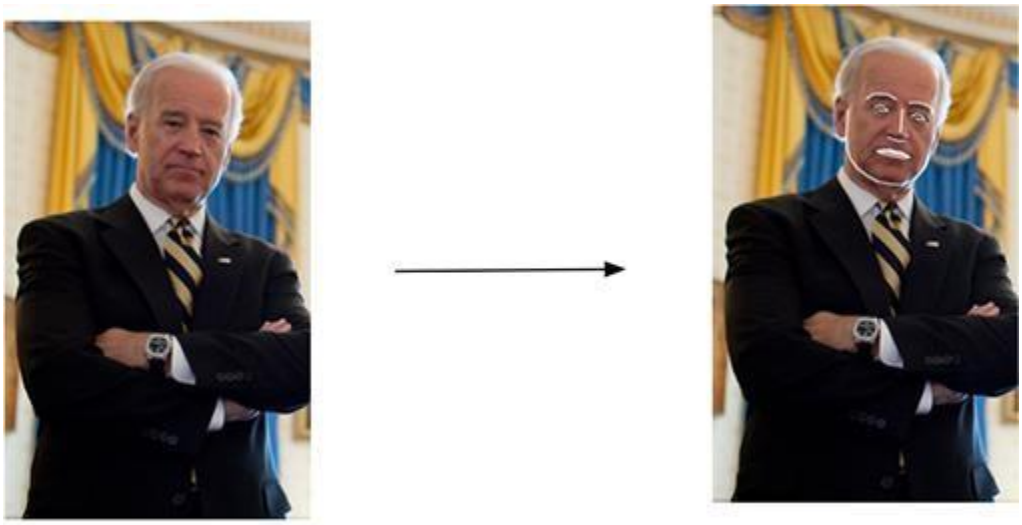


Input Output

```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```

Find and manipulate facial features in pictures

Get the locations and outlines of each person's eyes, nose, mouth, and chin.



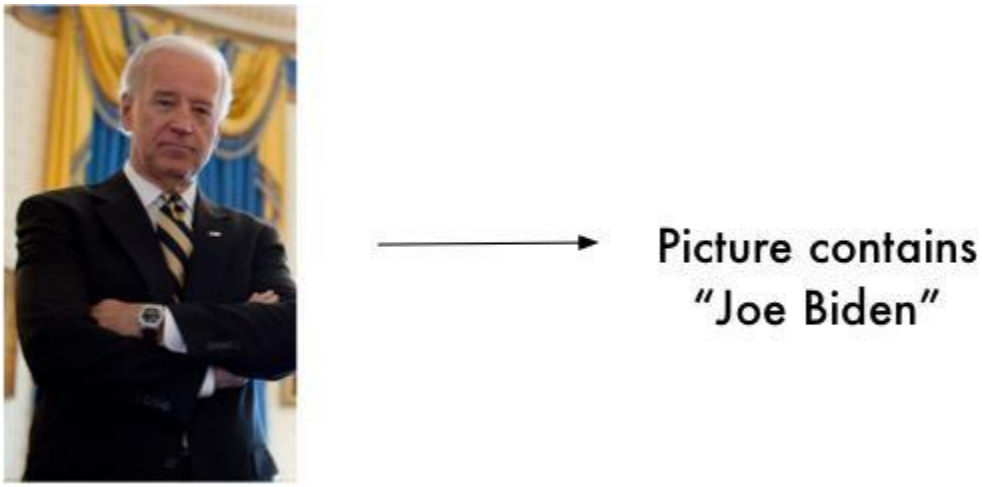
Input Output

```
import face_recognition
```

```
image = face_recognition.load_image_file("your_file.jpg")
face_landmarks_list = face_recognition.face_landmarks(image)
```

Identify faces in pictures

Recognize who appears in each photo.




```
import face_recognition
known_image = face_recognition.load_image_file("biden.jpg")
unknown_image =
face_recognition.load_image_file("unknown.jpg")

biden_encoding =
face_recognition.face_encodings(known_image)[0] unknown_encoding =
face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([biden_encoding],
unknown_encoding)
```

You can even use this library with other Python libraries to do real-time face recognition!

How Face Recognition works:

face recognition is really a series of several related problems:

1. First, look at a picture and find all the faces in it
2. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person.
3. Third, be able to pick out unique features of the face that you can use to tell it apart from other people— like how big the eyes are, how long the face is, etc.
4. Finally, compare the unique features of that face to all the people you already know to determine the person's name.

Pretty simple, right. Well as a human, your brain is wired to do all of this automatically and instantly. In fact, humans are too good at recognizing faces and end up seeing faces in everyday objects.

Let's tackle this problem one step at a time. For each step, we'll learn about a different machine learning algorithm:

Step 1: Finding all the Faces

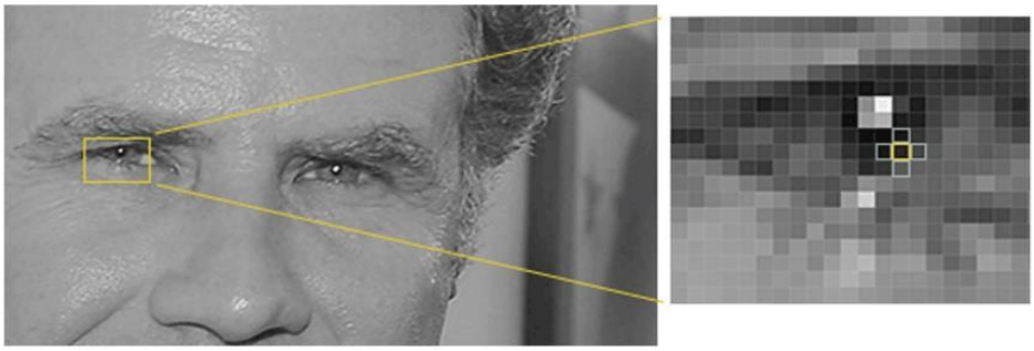
The first step in our pipeline is *face detection*. Obviously, we need to locate the faces in a photograph before we can try to tell them apart!

If you've used any camera in the last 10 years, you've probably seen face detection in action.

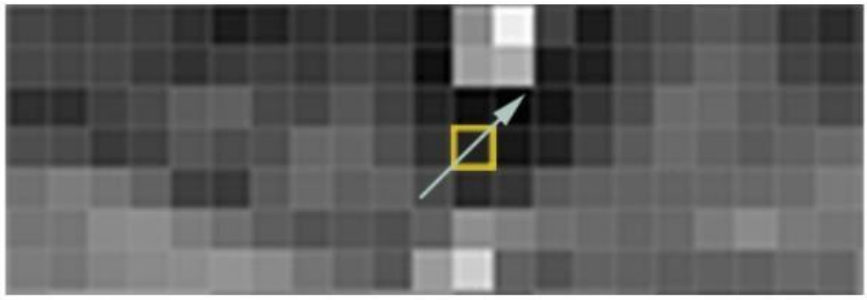
Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. We're going to use a method invented in 2005 called Histogram of Oriented Gradients — or just ***HOG*** for short.

To find faces in an image, we'll start by making our image black and white because we don't need color data to find faces:

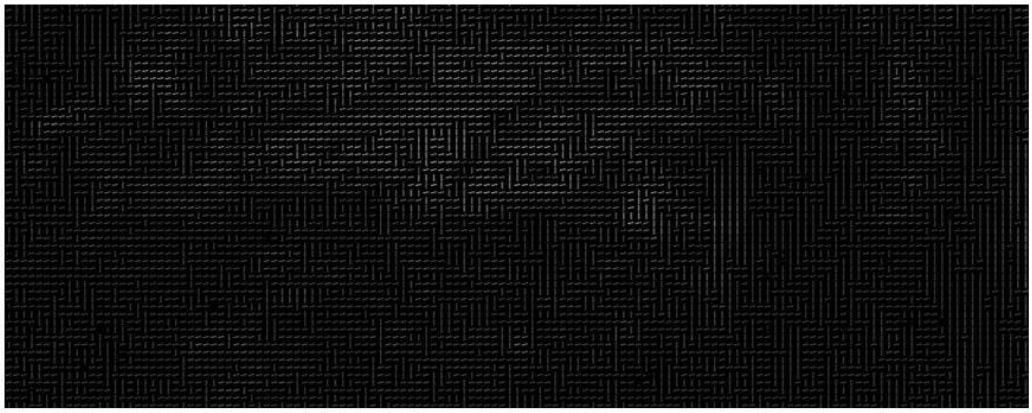
Then we'll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it:



Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:



If you repeat that process for **every single pixel** in the image, you end up with every pixel being replaced by an arrow. These arrows are called *gradients* and they show the flow from light to dark across the entire image:

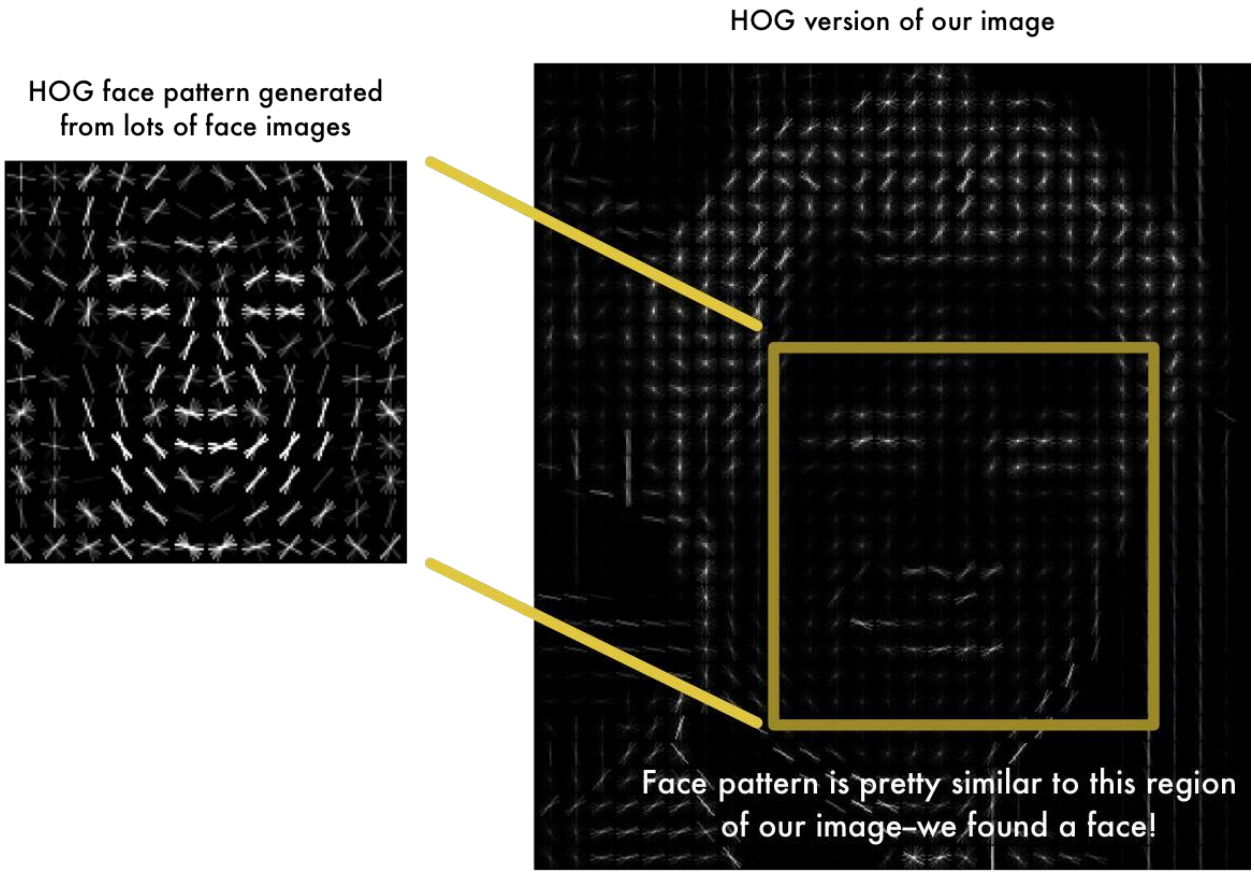


This might seem like a random thing to do, but there's a really good reason for replacing the pixels with gradients. If we analyze pixels directly, really dark images and really light images of the same person will have totally different pixel values. But by only considering the *direction* that brightness changes, both really dark images and really bright images will end up with the same exact representation. That makes the problem a lot easier to solve!

But saving the gradient for every single pixel gives us way too much detail. We end up missing the forest for the trees. It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image.

To do this, we'll break up the image into small squares of 16x16 pixels each. In each square, we'll count up how many gradients point in each major direction (how many points up, point up-right, point right, etc.…). Then we'll replace that square in the image with the arrow directions that were the strongest.

To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces:



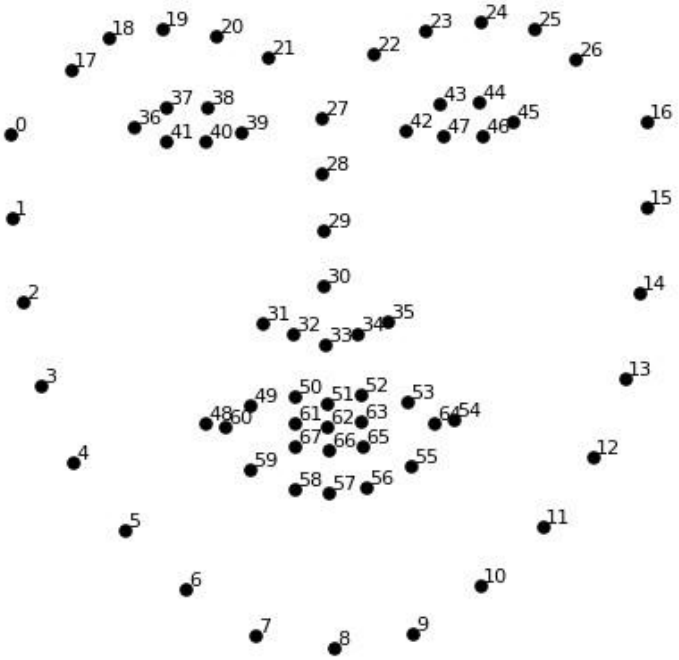
Step 2: Posing and Projecting Faces

Whew, we isolated the faces in our image. But now we have to deal with the problem that faces turned different directions look totally different to a computer.

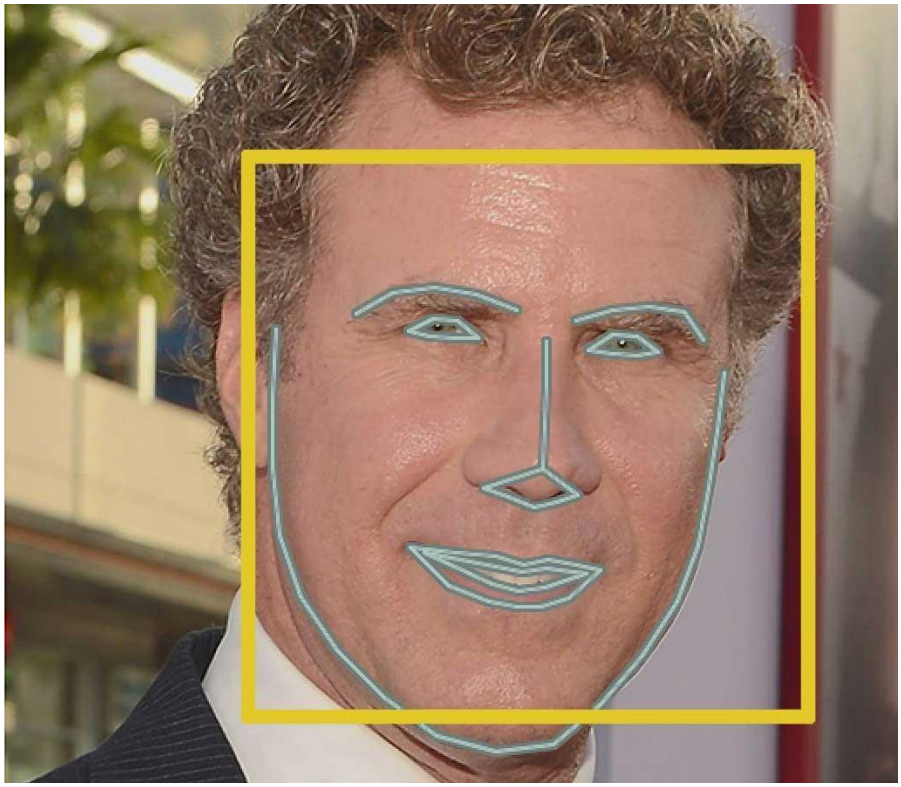
To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps.

To do this, we are going to use an algorithm called **face landmark estimation**. There are lots of ways to do this, but we are going to use the approach invented in 2014 by Vahid Kazemi and Josephine Sullivan.

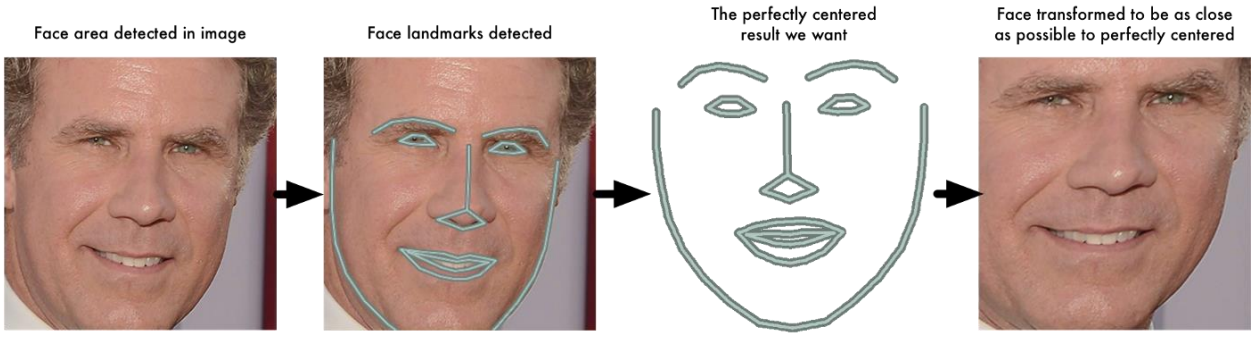
The basic idea is we will come up with 68 specific points (called *landmarks*) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face:



Here’s the result of locating the 68 face landmarks on our test image:



Now that we know where the eyes and mouth are, we'll simply rotate, scale and shear the image so that the eyes and mouth are centered as best as possible. We won't do any fancy 3d warps because that would introduce distortions into the image. We are only going to use basic image transformations like rotation and scale that preserve parallel lines (called affine transformations)



Now no matter how the face is turned, we are able to center the eyes and mouth are in roughly the same position in the image. This will make our next step a lot more accurate.

Step 3: Encoding Faces

Now we are to the meat of the problem — actually telling faces apart.
This is where things get really interesting!

The simplest approach to face recognition is to directly compare the unknown face we found in Step 2 with all the pictures we have of people that have already been tagged. When we find a previously tagged face that looks very similar to our unknown face, it must be the same person. Seems like a pretty good idea, right?

There's actually a huge problem with that approach. A site like Facebook with billions of users and a trillion photos can't possibly loop through every previous-tagged face to compare it to every newly uploaded picture. That would take way too long. They need to be able to recognize faces in milliseconds, not hours.

What we need is a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements.

which measurements should we collect from each face to build our known face database?

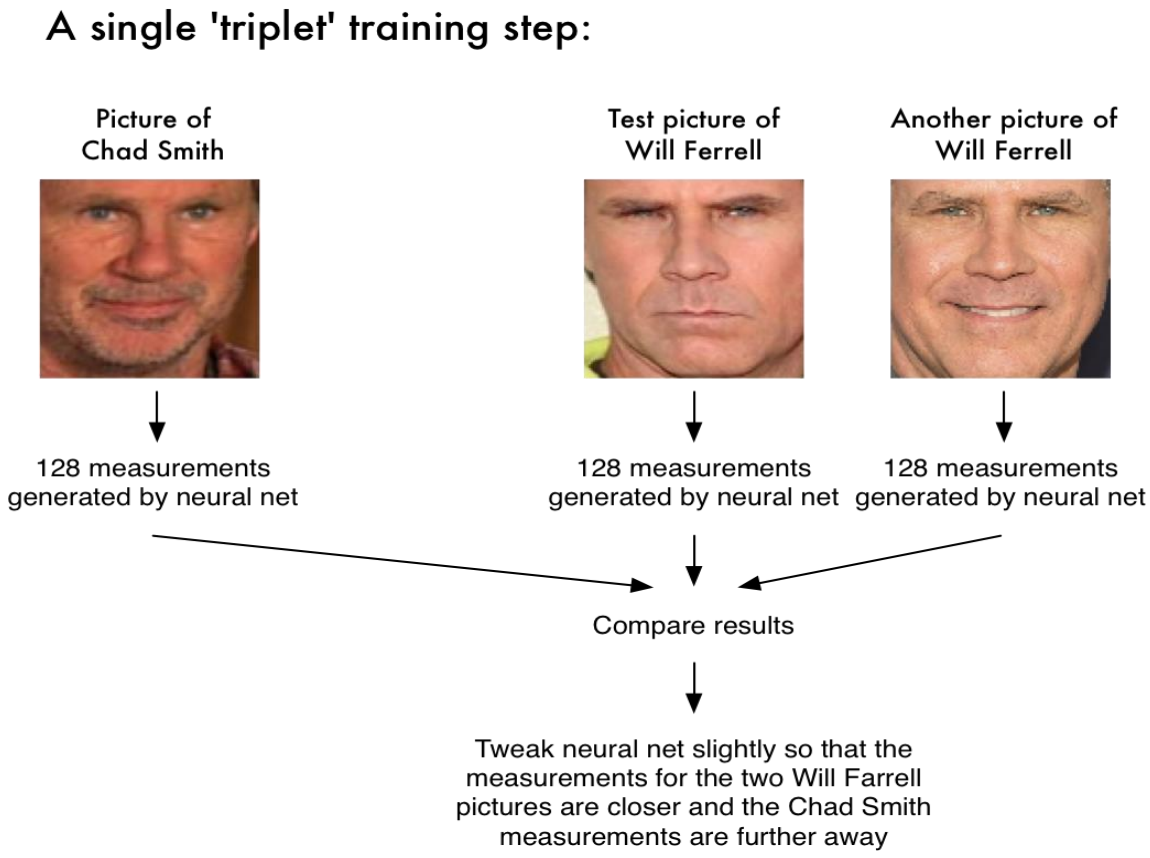
Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Network (just like we did in Part 3). But instead of training the network to recognize pictures objects like we did last time, we are going to train it to generate 128 measurements for each face.

The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person
2. Load another picture of the same known person
3. Load a picture of a totally different person

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart:



After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements.

Machine learning people call the 128 measurements of each face an **embedding**. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning (especially in language translation). The exact approach for faces we are using was invented in 2015 by researchers at Google but many similar approaches exist.

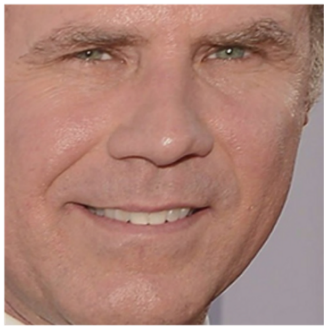
Encoding our face image

This process of training a convolutional neural network to output face embeddings requires a lot of data and computer power. Even with an expensive NVidia Tesla video card, it takes about 24 hours of continuous training to get good accuracy.

But once the network has been trained, it can generate measurements for any face, even ones it has never seen before! So, this step only needs to be done once. Lucky for us, the fine folks at OpenFace already did this and they published several trained networks which we can directly use. Thanks Brandon Amos and team!

So, all we need to do ourselves is run our face images through their pretrained network to get the 128 measurements for each face. Here's the measurements for our test image:

Input Image



128 Measurements Generated from Image

0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334968328	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

Step 4: Finding the person's name from the encoding

This last step is actually the easiest step in the whole process. All we have to do is find the person in our database of known people who has the closest measurements to our test image.

You can do that by using any basic machine learning classification algorithm. No fancy deep learning tricks are needed. We'll use a simple linear SVM classifier, but lots of classification algorithms could work.

All we need to do is train a classifier that can take in the measurements from a new test image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name of the person!

3.4 Software & Hardware Implementation:

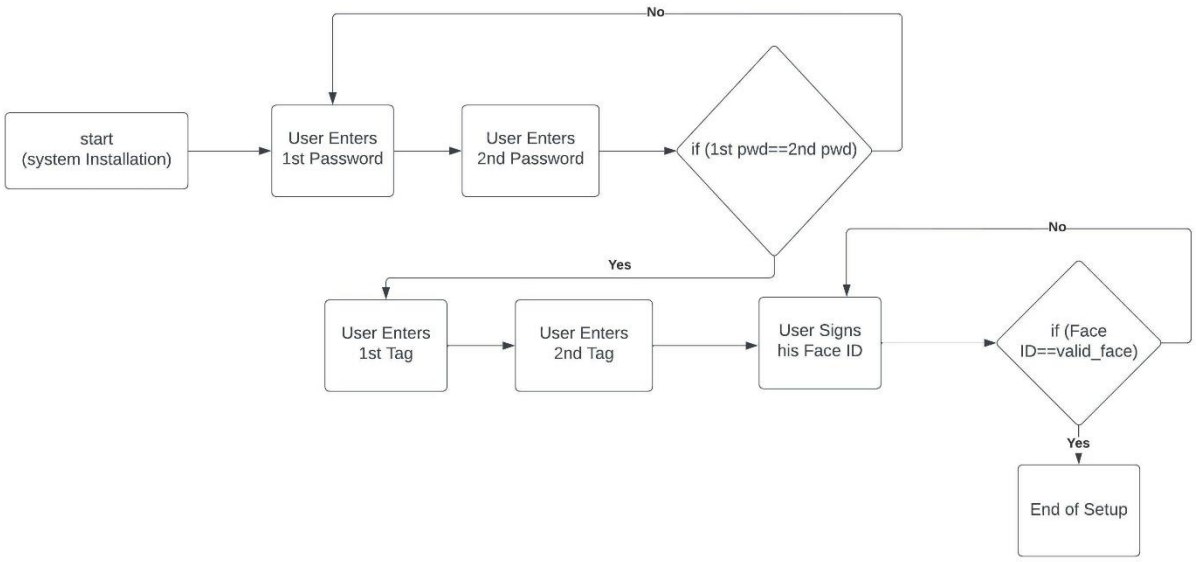


Figure 3. 24 System Setup

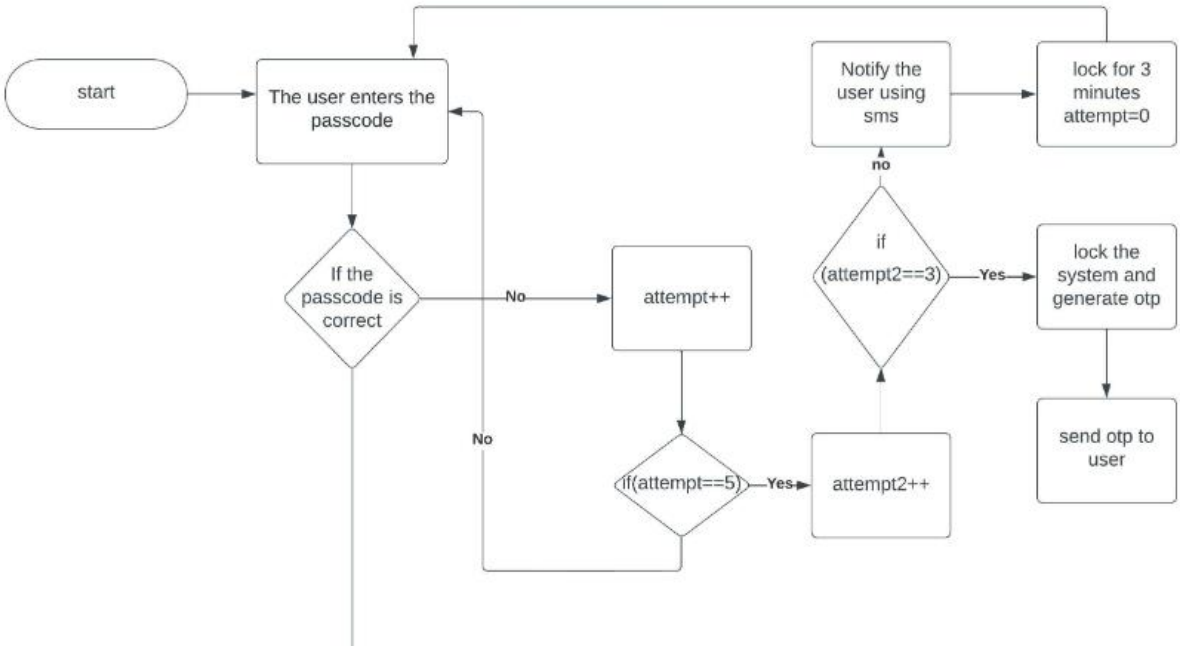


Figure 3. 25

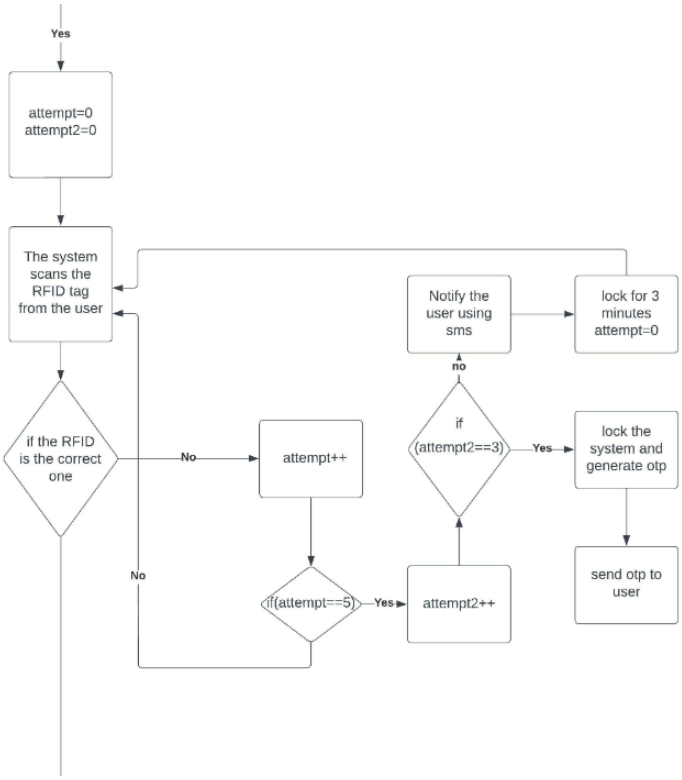


Figure 3. 26

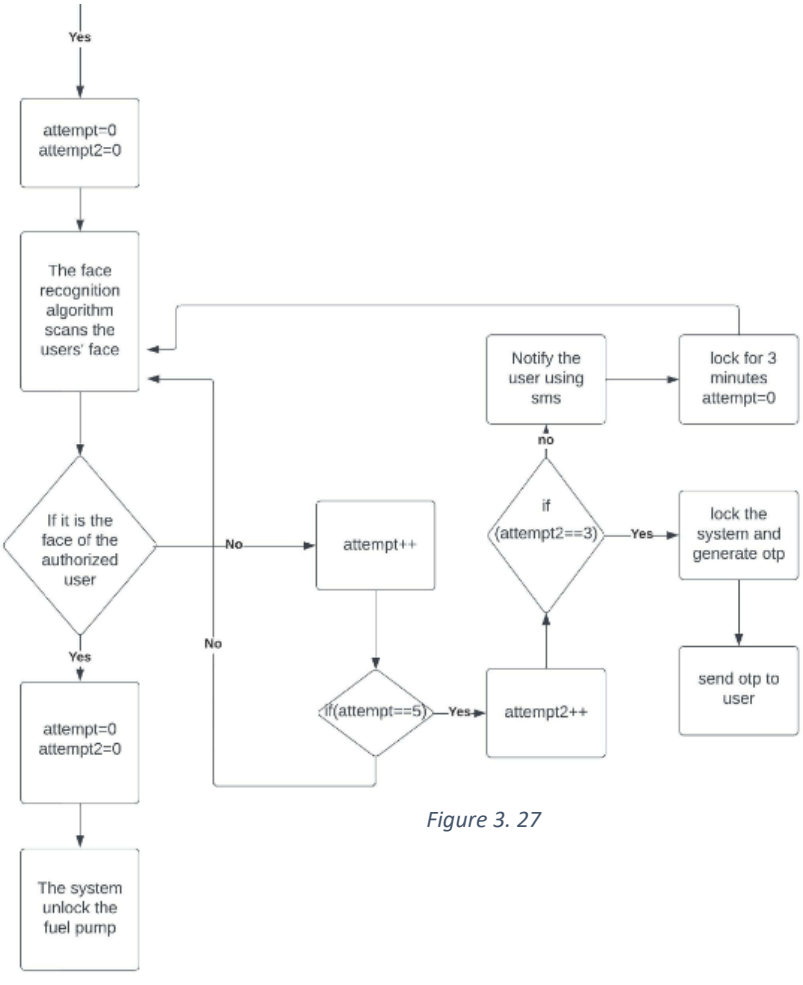


Figure 3. 27

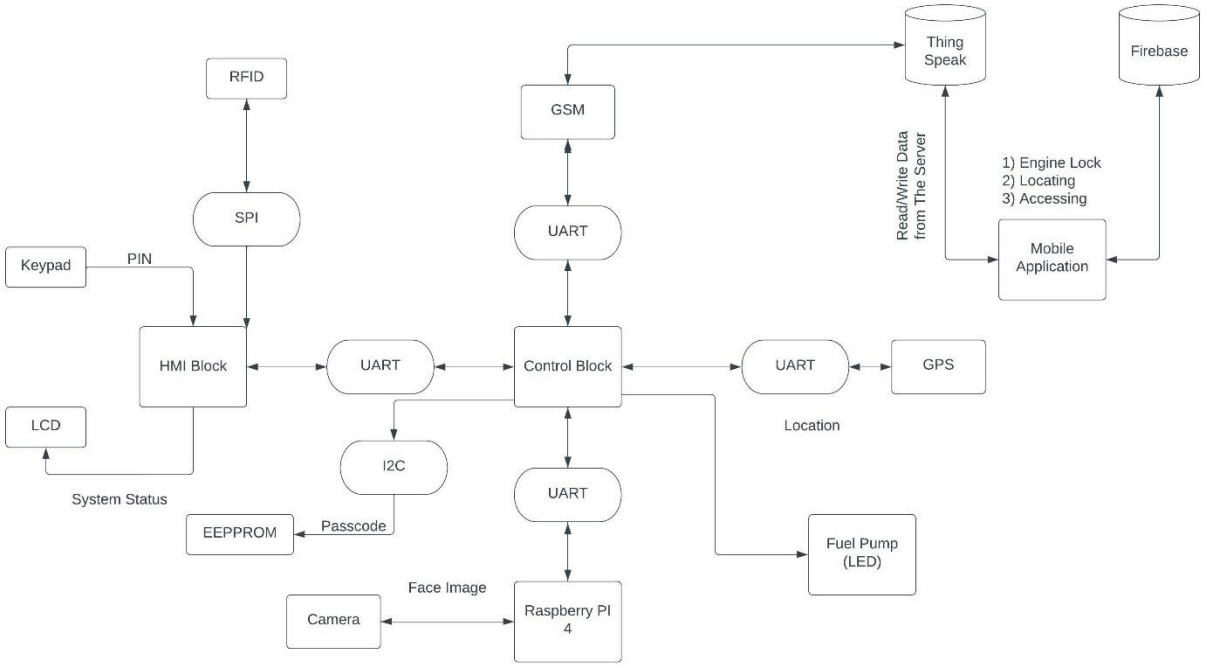


Figure 3. 28 Block Diagram

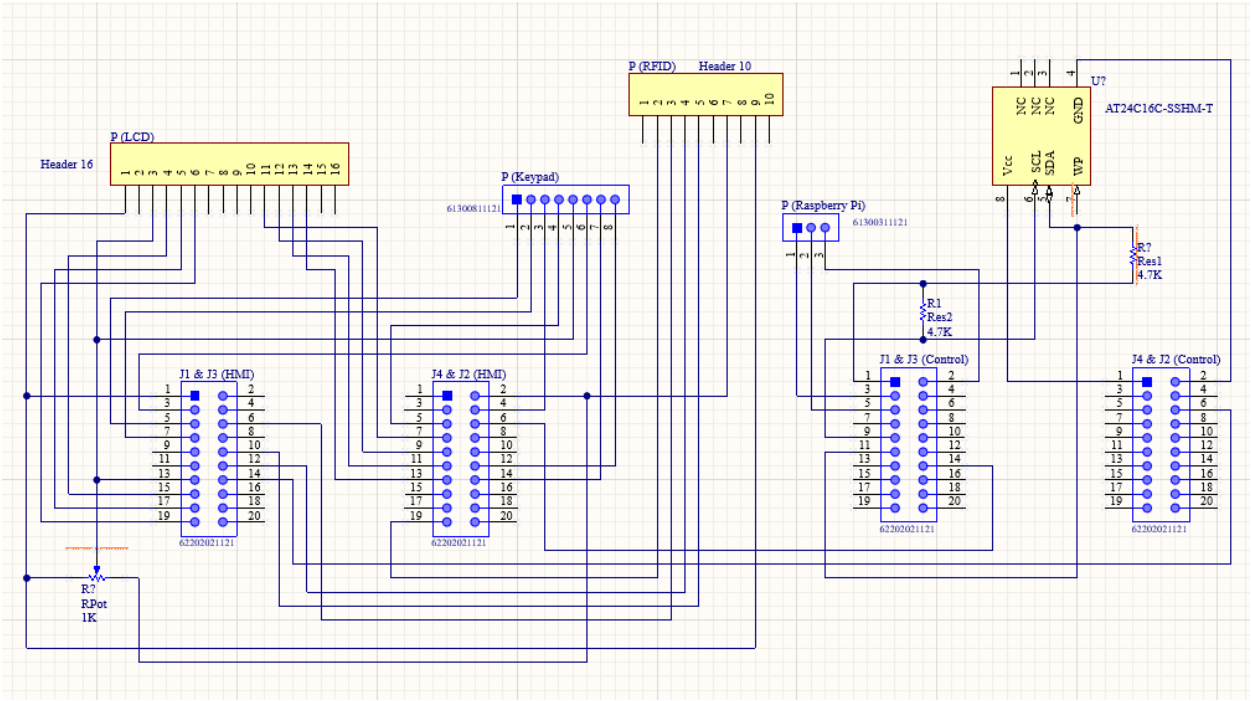


Figure 3. 29 System schematics

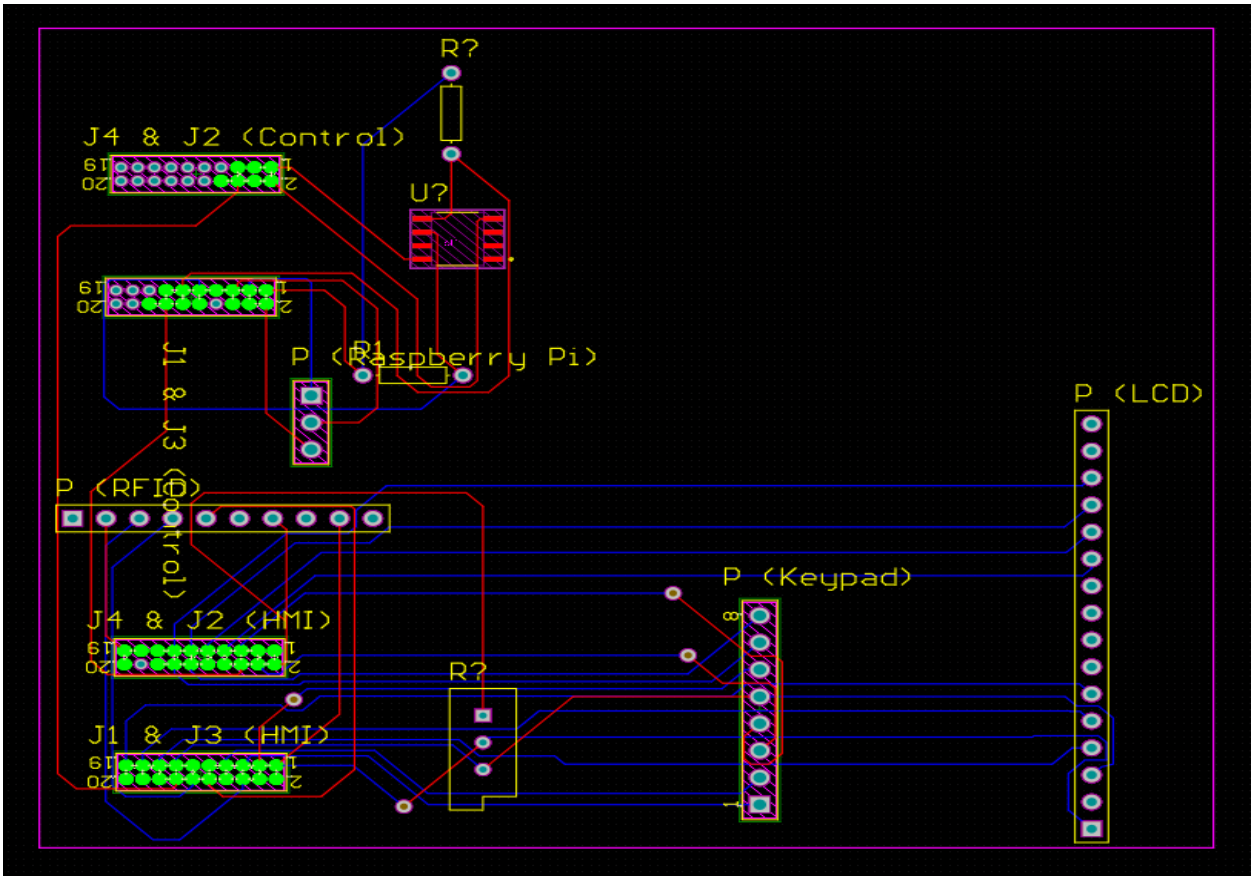


Figure 3. 30 PCB

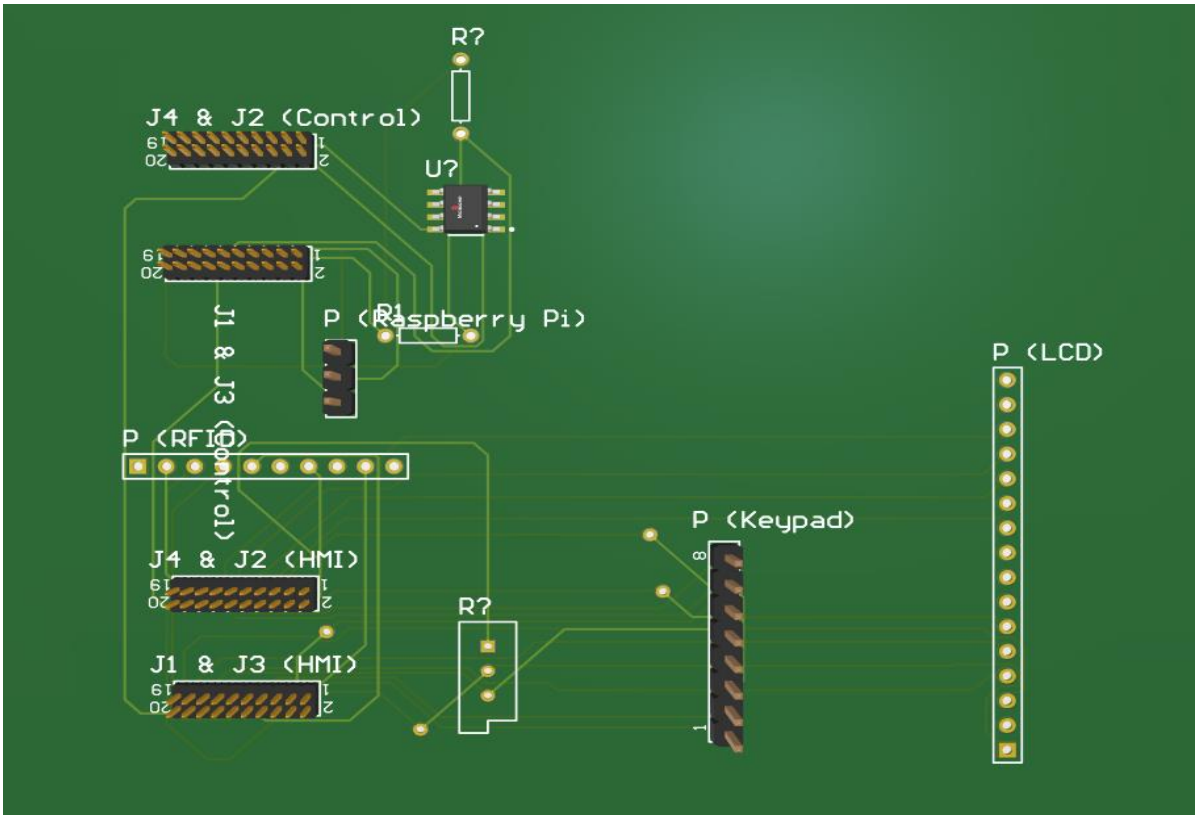


Figure 3. 31

We integrated Keypad, LCD, RFID, and camera which is connected to the Raspberry Pi, in authentication process which the user is required to go through in order to unlock the engine. Where the first three components are connected to the HMI block (Tiva C).

While the other Tiva C (Control Block) is connected to HMI block and Raspberry Pi through UART protocol and to EEPROM through i2c protocol which is used by the system to store the passcode and the number of attempts to access the system. The control block is also connected to GPS/GSM Module through UART where the GPS is used to find the location and GSM is used to communicate with the mobile application and write the location given by the GPS on the online server (ThingSpeak).

We have some steps the user must make in order to use the system.

First time process: After installing the hardware system in the vehicle and booting it for the first time the user is required to enter the passcode, scan RFID tag, and add their face ID, by interacting with the LCD, keypad, RFID reader (all the previous peripherals are connected to HMI Tiva C, and camera (connected with raspberry Pi)

They also should use the Android Application to be able to remotely control the system. They are required to sign up before using the application. Firebase database is used in this process to hold the users' data.

Now the system is ready to go.

Authentication: When the user tries to access the vehicle, they have to go through the authentication process. They have to enter their previously entered credentials.

If the authentication succeeds the fuel pump relay is unlocked and the vehicle is ready to go.

But if the authentication fails, the user can have up to 15 trials with 3 minutes of lock down between every 5 attempts and the hardware notifies the user of an attempt through his mobile.

In case of 15 failed attempts the system is fully locked down but for accessing the location. And the hardware sends an OTP to the user, who is now required to enter that code in the hardware in order to unlock the system.

Controlling the pump: The system takes an input signal from the vehicle that indicates if the vehicle is on or off. Depending on the status of that signal the hardware takes certain actions.

If the input is low that means the engine is off, and a timer is set to 5 minutes. After the 5 minutes pass reauthentication is required, and if the engine is turned on before that no reauthentication is required. This works also if the user authenticates and does not turn on their vehicle.

GSM/GPS process: A timer is used to interrupt the system every 5 minutes, to update the location of on ThingSpeak through the means of GSM and GPS. The user can read the written location using the Android Application.

The GSM uses interrupts too, as it is always waiting for an input from the Android Application.

Android Application: The Android application is used to make the user have a certain degree of controllability over the hardware. As mentioned before the user has to sign up before using the application and they are required to login after every time they close the application for security purposes.

The application is used to turn on/off the fuel pump relay and to locate the vehicle reading the last updated location on ThingSpeak.

3.5 Mobile Application Implementation:

Most of the Hardware designs for the past 5 years is designed with an Mobile Application to make it easy for the users to interact with the hardware in a remote way so that they can have full control over the hardware, and our project is no different so we designed a Mobile Application for Android Devices to help the car owner to control the engine lock, monitor the cars location and etc...

Implementation:

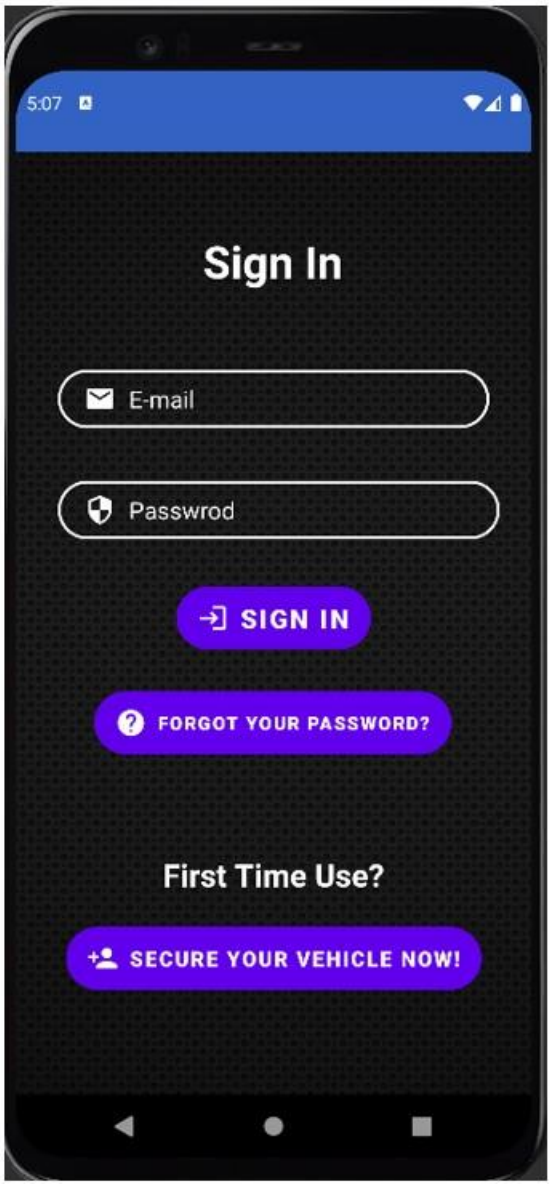
This Application was designed using Android Studio Software using both XML for the app's user interface and Java Programming Language to handle the Functionality of The Application like controlling the transition between the activity and the function of every button and Edit Text in each activity also **Firebase** and **Thing Speak** servers were used to hold the information given by the user, Firebase holds the personal information like username, email, password, chassis number while Thing Speak holds the location coordinates in Longitude and Latitude Form.

Activities:

Login Activity:

Login Activity is an essential for any mobile application as it distinguishes different users from each other and so that every user can have their own data and have the ability modify it whenever they want.

Screenshot of the Login Activity Screen:



Contents:

- E-mail Edit Text
- Password Edit Text
- Sign in Button
- Forgot Your Password? Button
- Secure your Vehicle Now! (Register) Button

Email Edit Text:

A user is required to enters his registered email on the data base in this textbox so that they can login into the app, otherwise if the user enters a wrong email or unregistered email a toast with the following text “Incorrect Email or Password” will appear on the screen or if the user enters unverified email address a toast with the following text “Please Verify your E-mail Address” on the screen and hence user is required to verify their email address to be able to login into the app.

Password Edit Text:

A user is required to enters their own password associated with the registered email saved on the data base in this textbox so that they can login into the app, otherwise if the user enters an incorrect password a toast with the following text “Incorrect Email or Password” will appear on the screen.

Sign in Button:

After the user enters his login credentials, he is required to press this button be login into the app if only the data they entered is valid the screen will change to Home Screen, if the user didn’t enter data and tried pressing the button a toast with the following text” Empty Credentials” will appear on the screen.

Forgot Your Password? Button:

If for some reason the user forgot their password, they can press this button which will change the screen to Reset Password Screen (Activity) and initiate the reset password sequence.

Secure your Vehicle Now! (Register) Button:

A user can use this button to register a new account for the application with his personal data including username, phone number, chassis number etc.…

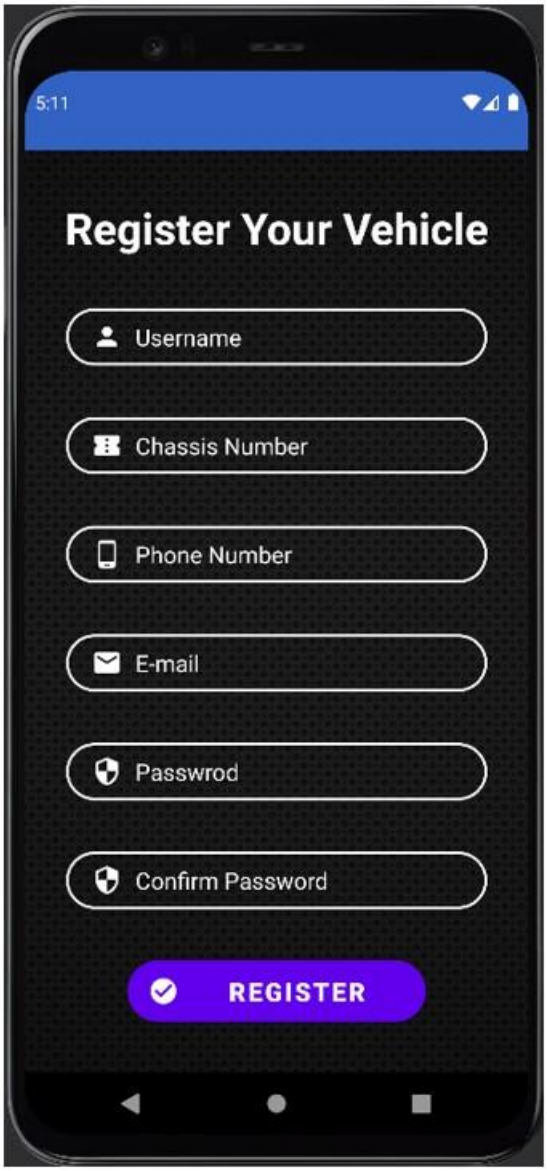
Register Activity:

Same as login activity register activity is essential in modern applications and websites as it's the way of saving your personal data including login credentials to a database in order for a system to distinguish you from the other users, the only way to get to this activity is press **Secure your Vehicle Now! Button** found in **Login Activity**.

Contents:

- Username Edit Text
- Chassis Number Edit Text
- Phone Number Edit Text
- Email Edit Text
- Password Edit Text
- Confirm Password Edit Text
- Register Button

Screenshot of the Register Activity Screen:



Username Edit Text:

The first thing in any registry form sometimes must be the username which acts as the user's representing name in the system, the user can choose any username for themselves and type it in this textbox.

Chassis Number Edit Text:

The Second and Probably the most important attribute is the Chassis Number as it acts as the distinguishing factor between vehicles as we know there are not two vehicles in the whole world with the same chassis number, so the user is required to enter his vehicle's chassis number in this text box.

Phone Number Edit Text:

in this textbox the user is required to enter their personal phone number.

Email Edit Text:

The user is required to enter their personal email in order to create the account but bear in mind that the user can only choose an email that hasn't been registered otherwise if the user enters a registered email a toast with the following text "User already exists on this E-mail" will pop u on the bottom side of the screen.

Password Edit Text:

The use should pick a password for their account to secure it, but this password needs to fulfill a condition in which the number of characters must not be less than 6 characters otherwise if the user enters less than 6 characters a toast with "Password is too short" will appear on the screen when the user presses register button.

Confirm Password Edit Text:

In this text box the user is required to enter the same password they entered in the password text box otherwise a toast with the following text "Password Mismatch" will appear on the screen.

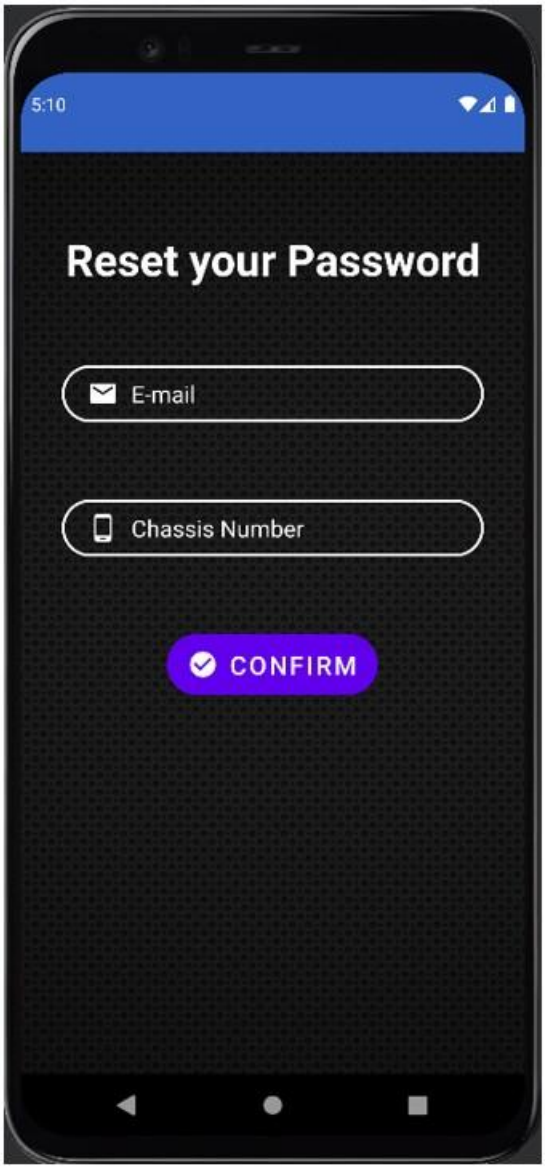
Register Button:

After the user fills all the other text boxes, they can just press register to continue to the verification activity to complete the registration process and a toast with the following text “Registration Succeeded” will appear on the screen, yet there few constraints in order for the app to switch to the verification activity like the password must be at least 6 characters and the user must fill every textbox otherwise a toast with the following text will appear “Empty Credentials”.

Reset Password Activity:

In this activity the user can reset their password in case they forgot it.

Screenshot of the Reset Password Activity Screen:



Contents:

- E-mail Edit Text
- Chassis Number Edit Text
- Confirm Button

E-mail Edit Text:

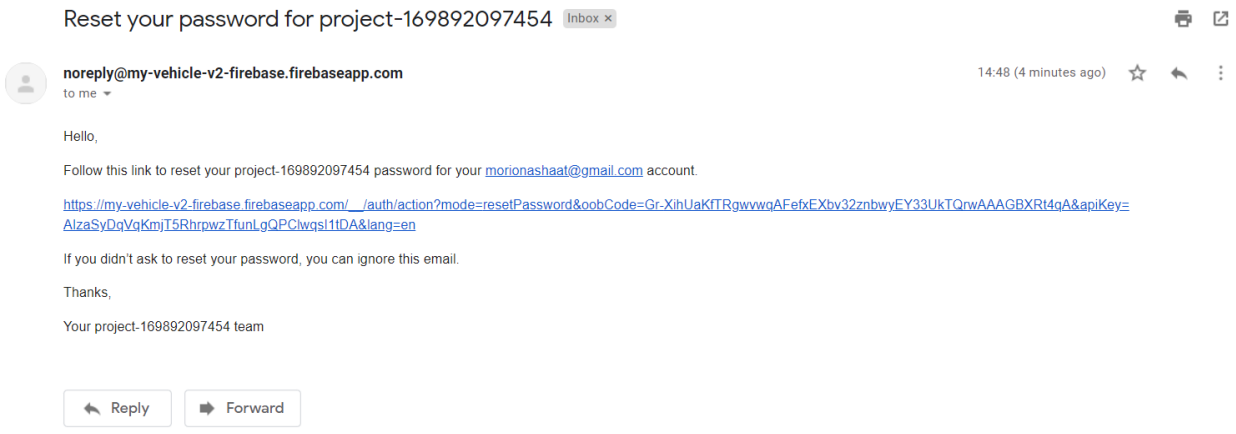
In this textbox the user must enter their registered email in case they want to reset the password, if the user enters an incorrect E-mail Address or unregistered a toast with the following text “User doesn’t exist” on the bottom side of the screen.

Chassis Number Edit Text:

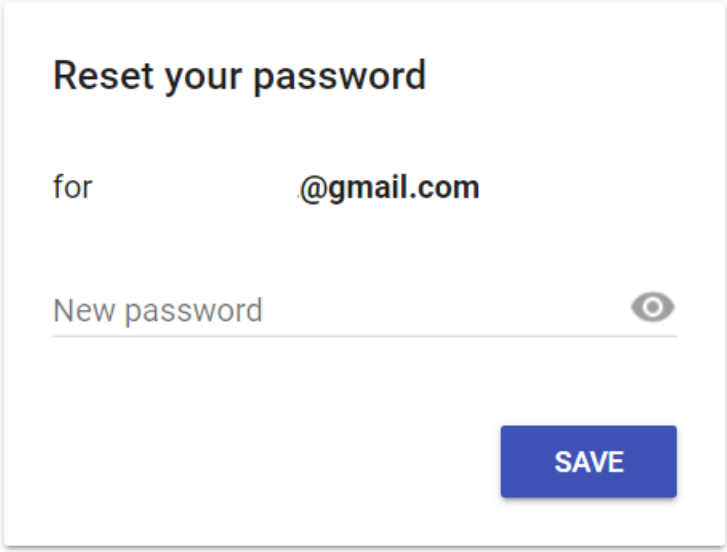
The user enters their registered chassis number on the database (Firebase) that matches with the entered email of their vehicle in this textbox.

Confirm Button:

After the user enters the required data for resetting the password, they can press confirm button and on success and the validation of data an email will be sent your email to create a new password as in the following figures and a toast with the following text ”Reset Link sent to email” will appear on the screen and in case the chassis number is incorrect a toast with following text “Wrong E-mail or Chassis Number” will appear on the screen.



After the user presses on the link, it transfers the user to this webpage where they can choose new password for their account.

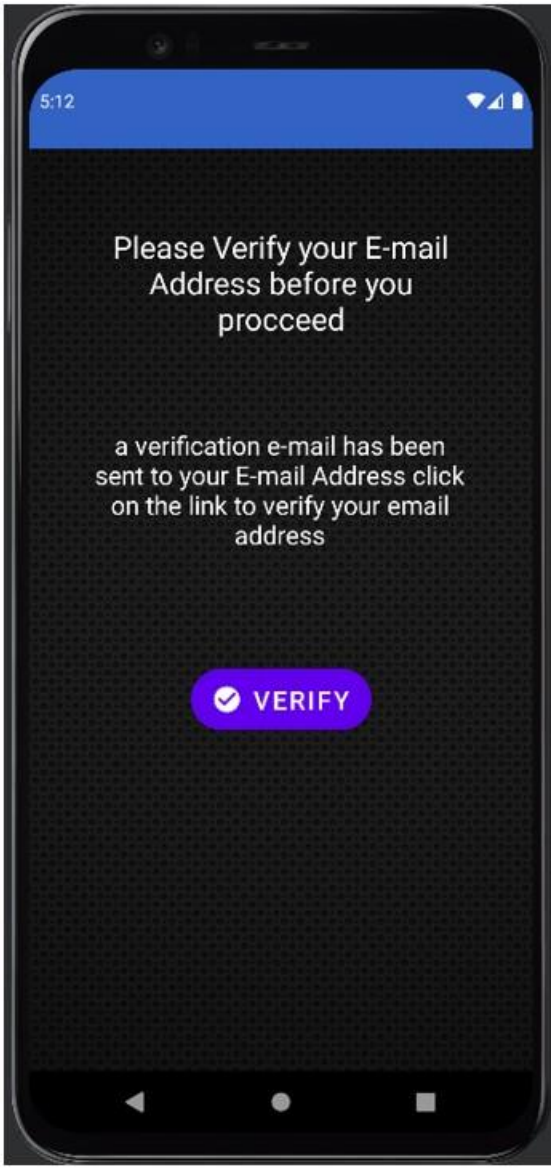


Then the user can now press save and start using the new password for logging into their account

Verification Activity:

In this activity the user needs to verify the email address they entered in the registration form in order to finalize the registration process and able to sign into their account.

Screenshot of the Verification Activity Screen:

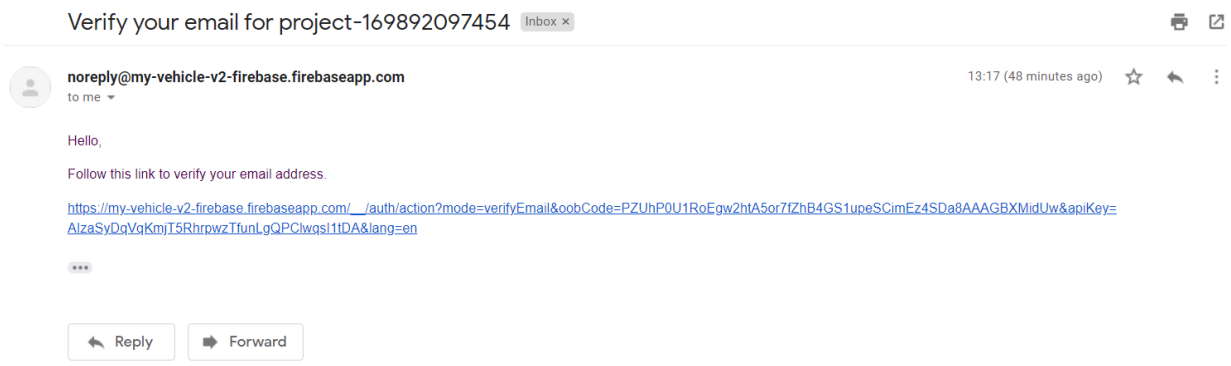


Contents:

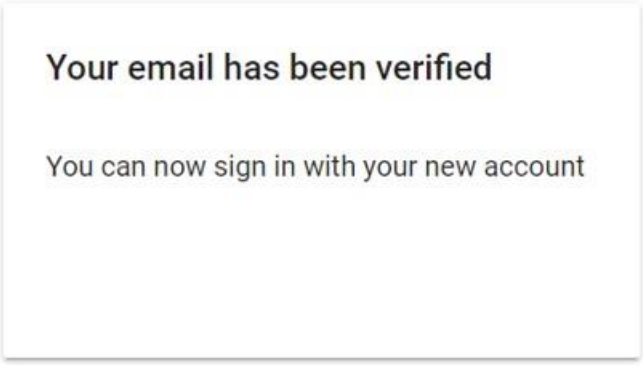
- Verify Button

Verify Button:

This single button actually does a lot of stuff, once the user presses verify a verification email is sent from the project's Firebase to the email you entered in the registration form like the following image:



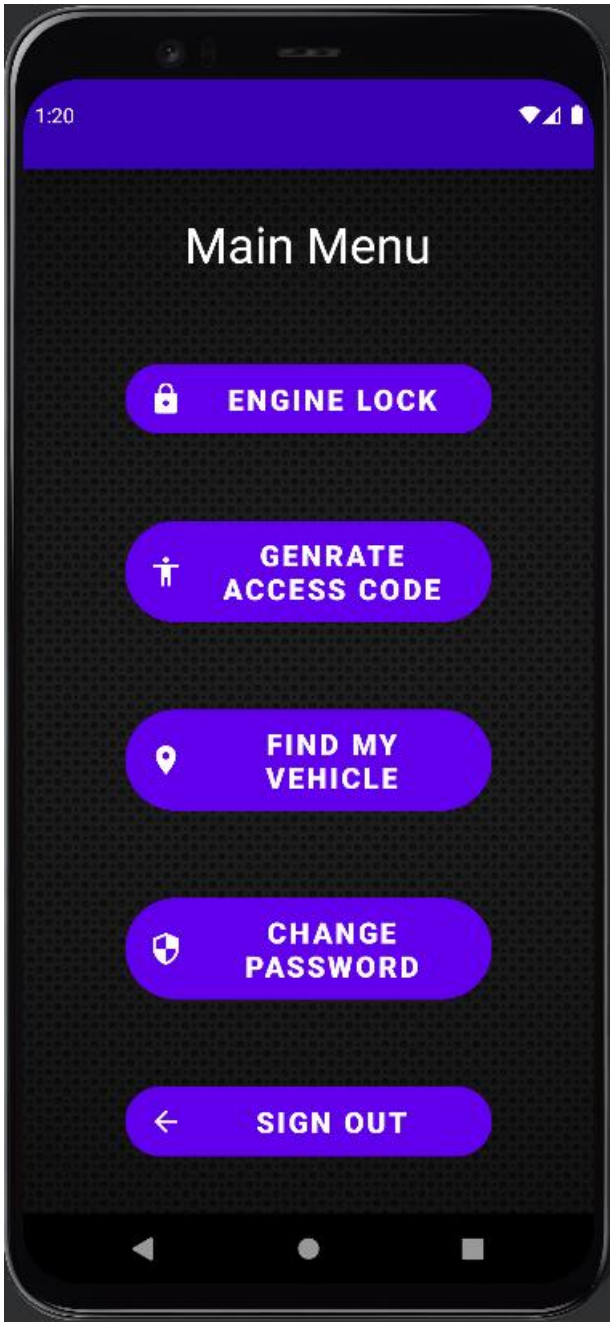
When the user presses on the link sent in the email it will transfer to the user to following page indication that the email has been verified and the user can now login into their email.



Home Activity:

Home activity is the activity where all the functionality is stored.

Screenshot of the Home Activity Screen:



Contents:

- Engine Lock Button
- Generate Access Code Button
- Find My Vehicle Button
- Sign Out button

Engine lock Button:

If the user wishes to lock the engine after the engine is turned on, the y can simply tab on this button, which send a code to the Hardware System, then the system checks on the code sent and switches off the relay connected with the fuel pump.

Generate access code Button:

This button can be used to generate a code which works for a while and bypasses the authentication process. This way the vehicle can be shared easily with more than one user.

Find my vehicle Button:

This button sends back the last location of the vehicle. When pressed the Hardware will receive a code and responds by sending the location of the vehicle through **ThingSpeak**, then the Android Application reads the location and asks the user to open a third-party application which is used to view the location fetched (i.e., Google Maps).

Screenshot of Google Maps Application with the location:

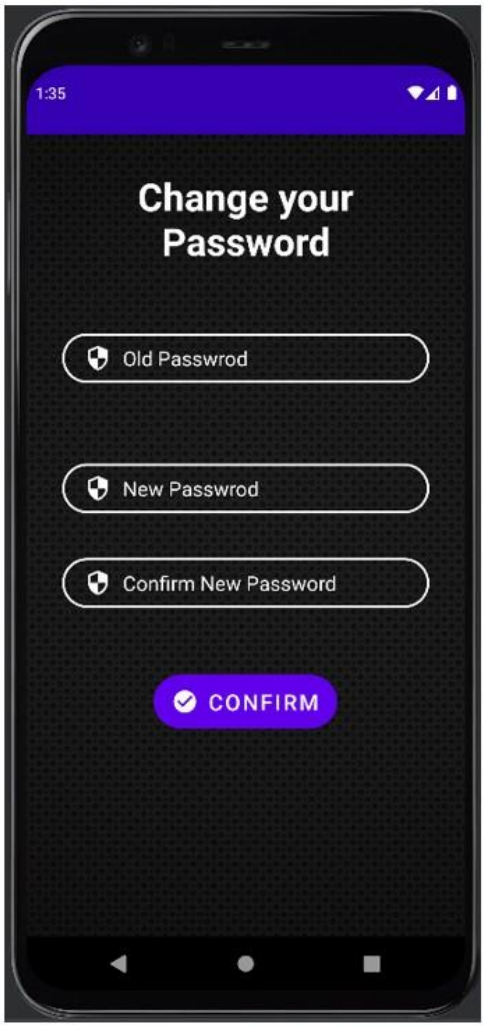
Sign out Button

The user can sign out by clicking on this button (obviously). This sends the user back to the Login Activity.

Change Password Activity:

This activity gives the user the ability to change their password whenever they like to, using only the old password.

Screenshot of the Change Password Activity Screen:



Contents:

- Old Password Edit Text
- New Password Edit Text
- Confirm New Password Edit Text
- Confirm Button

Old Password Edit Text:

In this textbox the user is required to enter their old password in order to be able to change their password.

New Password Edit Text:

In this textbox the user enters their new chosen password that should fulfill the same conditions as the old one which the number of characters must not be less than 6 characters otherwise if the user enters less than 6 characters a toast with “Password is too short” will appear on the screen when the user presses confirm button.

Confirm New Password Edit Text:

In this text box the user is required to enter the same password they entered in the password text box otherwise a toast with the following text “Password Mismatch” will appear on the screen.

Confirm Button:

The user presses this button to confirm the changed password in case of success operation a toast of the following text “Password Changed” will appear on the screen and the user can now login using the new password and in case the user entered wrong old password the following toast ”Wrong Old Password” will appear on the screen.

Chapter 4: Results and Discussion

Unfortunately, we were not able to test our system on a real car. So, we tested the system on simple LEDs and buzzers as to verify the authenticity of it.

Upon testing our system and comparing it to previously mentioned systems in chapter 2 we found the following (kindly refer to chapter 2 to see full review of these systems and follow their links):

1. FACE RECOGNITION BASED INTELLIGENT CAR ANTI-THEFT SYSTEM USING RASPBERRY PI, Driver Face Recognition: Anti-Theft System

   Comparing ours with those, we found that using facial recognition alone is not enough as it can be easily overridden using a photograph of the owner, that's why we added the other to validation methods, passcode and RFID.

2. Smart Vehicle Security and Defending Against Collaborative Attacks by Malware, Development of Smart Vehicle Security and Entertainment System (SSES) using Raspberry Pi, A Cost-Effective Method for Automobile Security Based on Detection and Recognition of Human Face, Face Recognition Based Car Ignition and Security System

   This time these systems added GSM and GPS modules and we found that was very useful that's why we talk similar measures in our system and added the two modules to make sure that the user can have a degree of remote control over the system

3. Microcontroller-based RFID, GSM and GPS for Motorcycle Security System, Anti-Theft System for Car Security using RFID, Smart Security System for Vehicles using Internet of Things (IoT)

   These systems added to the GSM and GPS modules the RFID module, which added a decent push to the security measures. We thought it was a very good addition, so we added RFID to ensure the security is of a high degree.

4. ANTI-THEFT VEHICLE SECURITY SYSTEM USING FINGERPRINT SCANNER AS WELL AS MANUAL, Fingerprint based Anti-Theft for Two Wheelers Authentication of Vehicle users

Fingerprint scanner was the main module in these systems. While it is secure, it suffers from fingerprint spoofing which makes it a hassle to use.

5. An Attempt to Develop an IOT based Vehicle Security System, Anti-Theft Security System for Vehicles

   These systems mentioned use password next to GPS and GSM modules. It is a simple security measure that can be bypassed easily, that's why passcode is vulnerable alone.

In the end we had a verry good idea of what makes a good security system. That's why we made a three-way authentication to make sure no one will access the vehicle except the authorized users. We also added GSM and GPS modules that send and receive data from the Android Application to help the user communicate with the hardware remotely.

Real Timing Implementation:

Process	Time (seconds) (Min-Max)
Sending SMS from hardware to mobile	2-5
Receiving SMS from mobile to hardware	3-6
Face Capture	3-4
Face Recognition	3-5

Chapter 5: Conclusions and Future Work

5.1. Conclusion

Car theft is huge problem that needs addressing, specially to those who cannot afford luxurious cars with sophisticated security systems. That's why we proposed a system to help car owners achieve their security needs with an affordable price.

In chapter 2 we made a full review of some systems, and we used some of the information in these systems. We also added to those systems as we implemented multiple validations, with remote access as well.

After integrating all the parts of software and hardware, we managed to achieve a decent degree of security to ensure that the vehicle will not be stolen, as we used three ways of authentication that the user has to go through to be able to start the engine of the vehicle.

1. Passcode
2. RFID
3. Face Recognition

The hardware system outputs whatever the user is required to do on the LCD, be it entering a passcode, using RFID tag, or use the camera to start the face recognition algorithm.

The user can also locate the vehicle using a remote Android Application with the means of both GSM and GPS modules, hence the user has a remote access on the security system.

The Android Application uses Firebase to save the users' login data and uses ThingSpeak to read the location (longitude and latitude) which the hardware has written.

- **The only requirement is that the vehicle has a fuel pump, which is found in most vehicles. This targets low end automobiles as well as new ones!**

5.2. Future Work

Our system is generally good, but not perfect! It can use some modifications to facilitate the remote access as well as additions to

1. Track the location of the vehicle instead of simply locating it.

2. Increase the users' degree of controlling the system.

3. The system could use a close-range (Bluetooth) detector to wake the system when the user approaches.

4. Add a rechargeable battery, which charges itself when the vehicle is on using the vehicle's battery.

5. Add deep learning algorithms, so that it can gather enough data of the user and use that data to facilitate the authentication process.

6. Enhance the system so that it can have more control on the vehicle, as to use the Android Application to even replace the key fob.

7. In case of unauthorized access, the system can call the authorities automatically and send the location, too.

8. Optimize power consumption with sleep mode

References

1. Fürst, S., Mössinger, J., Bunzel, S., Weber, T., Kirschke-Biller, F., Heitkämper, P., ... & Lange, K. (2009, October). AUTOSAR–A Worldwide Standard is on the Road. In *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden* (Vol. 62, p. 5).

2. Fürst, S., & Bechter, M. (2016, June). AUTOSAR for connected and autonomous vehicles: The AUTOSAR adaptive platform. In *2016 46th annual IEEE/IFIP international conference on Dependable Systems and Networks Workshop (DSN-W)* (pp. 215-217). IEEE.

3. Kum, D., Park, G. M., Lee, S., & Jung, W. (2008, October). AUTOSAR migration from existing automotive software. In *2008 International Conference on Control, Automation and Systems* (pp. 558-562). IEEE.

4. Specification of DIO Driver AUTOSAR CP R19-11, Last accessed 14-6-22

5. Specification of Port Driver AUTOSAR CP R19-11, Last accessed 14-6-22

6. adre, D. V., & Gupta, S. (2018). Tiva C Series Based Standalone Projects. In *Getting Started with Tiva ARM Cortex M4 Microcontrollers* (pp. 261-270). Springer, New Delhi.

7. Gadre, D. V., & Gupta, S. (2018). Tiva C Series Microcontroller Breakout Board. In *Getting Started with Tiva ARM Cortex M4 Microcontrollers* (pp. 53-60). Springer, New Delhi.

8. Nanda, U., & Pattnaik, S. K. (2016, January). Universal asynchronous receiver and transmitter (uart). In *2016 3rd international conference on advanced computing and communication systems (ICACCS)* (Vol. 1, pp. 1-5). IEEE.

9. Mankar, J., Darode, C., Trivedi, K., Kanoje, M., & Shahare, P. (2014). Review of I2C protocol. *International Journal of Research in Advent Technology*, *2*(1).

10. Gadre, D. V., & Gupta, S. (2018). Serial Communication: SPI and I2C. In *Getting Started with Tiva ARM Cortex M4 Microcontrollers* (pp. 211-238). Springer, New Delhi.

11. ADuCM3029 data sheet. Analog Devices, Inc., March 2017.

12. Nugent, Stephen. "Precision SPI Switch Configuration Increases Channel Density." *Analog Dialogue*, May 2017.

13. Usach, Miguel. AN-1248 Application Note: *SPI Interface*. Analog Devices, Inc., September 2015.

14. Dhaker, P. (2018). Introduction to SPI interface. *Analog Dialogue*, *52*(3), 49-53.

15. Gadre, D. V., & Gupta, S. (2018). Interrupts. In Getting Started with Tiva ARM Cortex M4 Microcontrollers (pp. 123-130). Springer, New Delhi.

16. Morales, M. (2013). An Introduction to the Tiva™ C Series Platform of Microcontrollers. Texas Instruments, April.

17. Gadre, D. V., & Gupta, S. (2018). ARM Cortex-M4 Core and Tiva C Series Peripherals. In Getting Started with Tiva ARM Cortex M4 Microcontrollers (pp. 13-26). Springer, New Delhi.

18. Bai, Y., & Roth, Z. S. (2019). Introduction to Tiva C MCU LaunchPad™—TM4C123G. In *Classical and Modern Controls with Microcontrollers* (pp. 35-99). Springer, Cham.

19. Tiva™ TM4C123GH6PM Microcontroller datasheet https://www.ti.com/lit/ds/spms376e/spms376e.pdf?ts=1655220348731&ref_url=https%253A%252F%252Fwww.google.com%252F

20. Raspberry Pi 4 Model B datasheet https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

21. Raspberry Pi NoIR Camera v2 datasheet https://www.farnell.com/datasheets/2056180.pdf

22. Juels, A. (2006). RFID security and privacy: A research survey. *IEEE journal on selected areas in communications*, *24*(2), 381-394.

23. MFRC522 Standard performance MIFARE and NTAG frontend

24. 4x4 Matrix Membrane Keypad datasheet, last accessed 3-5-22

25. Al-Khedher, M. A. (2012). Hybrid GPS-GSM localization of automobile tracking system. *arXiv preprint arXiv:1201.2630*.

26. Bharavi, U., & Sukesh, R. M. (2017, May). Design and development of GSM and GPS tracking module. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)* (pp. 283-288). IEEE.

27. Adam Geitgey. 2017. Face_recognition. ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line (github.com)