Ministry of Communications
and Information Technology

Professional Training Programs on Communications and Information Technology

(3 Months)

Track Title:

Embedded System (Idea to Product)

August 2020

Driver Assistance System

(DAS)

Prepared By:

Abdallah Ragab Mohamed

Ahmed Hossam El-Din Safaa

Hossam Mohamed Farouk

Mohamed Abdelkader Mostafa

Moustafa Raafat Moustafa

Supervised By:

Dr. Ghazal Fahmy

Dr. Mohamed El-Zorkany

Dr. Mostafa Shiple

Eng. Mahmoud Hussien

Eng. Nagdy Abd El-Hameed

Acknowledgement

This work represents the full report of graduation project 2020 of our group at National Telecommunications Institute, Embedded system - From Idea to Product track.

This work is the quintessence of continuous work and research at the department of electronics at NTI under of the supervision of the prestigious staff it has there to them we give our thanks.

Achievement of this work has been one of the most momentous challenges we have ever had to face. Without the support, patience and guidance of the following people, this work would not have been completed. We are greatly indebted to everyone for helping us to finalize this work.

First and foremost, we are thankful to our supervisors Dr. Ghazal, Dr. Mohammed El-Zorkany and Dr. Mostafa Shiple, whose guidance, fruitful discussion, suggestions and encouragement from the preliminary until the day of oral discussion enabled us to understand and develop the subject.

Finally, thanks to the assistant teachers and technicians especially Eng. Mahmoud Hussein and Eng. Nagdy in the electronics department for their assistance during the practical work.

Abstract

Driver Assistance Systems are intelligent systems that reside inside the vehicle and assist the main driver in a variety of ways.

The main aim that we have is to create an assistance system to help the driver during driving the car and create an automated car parking system with minimal human interference and remotely system start.

Before we begin talking working on the product, we decided to make a SWOT analysis of our product and the idea.

Strengths:
- **Low cost.**
- **Light adjustment.**
- **Time saving.**
- **Mobile App instead of Remote.**
- **Less stressful driving experience.**

Weaknesses:
- **Not so accurate data as cheaper sensors were used.**

Opportunities:
- **As the field of compound building is increasing, new infrastructure can open a huge market for this system.**
- **System can be added to any vehicle not designed for a specific one.**

Threats:
- **Infrastructure is not suitable in all places.**
- **Misuse of technology.**

Contents

Table of Figures

Chapter 1

Introduction

The main approach of this chapter is to give a brief introduction to the two main technologies which this project is based on (Embedded system, Real Time Operating System, and Internet of Things).

Embedded Systems

An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is designed for a specific function or for specific functions within a larger system. Industrial machines, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys as well as mobile devices are all possible locations for an embedded system.

Embedded systems are computing systems, but can range from having no user interface (UI) -- for example, on devices in which the embedded system is designed to perform a single task -- to complex graphical user interfaces (GUI), such as in mobile devices. User interfaces can include buttons, LEDs, touchscreen sensing and more. Some systems use remote user interfaces as well.

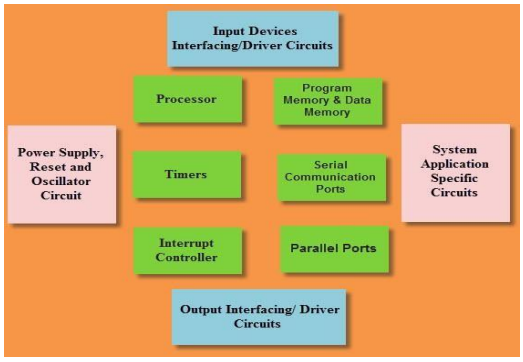


FIGURE 1: MICROCONTROLLER COMPONENTS

Embedded Systems Hardware

Embedded systems can be microprocessor or microcontroller based. In either case, there is an integrated circuit (IC) at the heart of the product that is generally designed to carry out computation for real-time operations.
Microprocessors are visually indistinguishable from microcontrollers, but whereas the microprocessor only implements a central processing unit (CPU) and thus requires the addition of other components such as memory chips, microcontrollers are designed as self-contained systems.
Microcontrollers include not only a CPU, but also memory and peripherals such as flash memory, RAM, or serial communication ports.
Because microcontrollers tend to implement full (if relatively low computer power) systems, they are frequently use on more complex tasks.
Microcontrollers are used, for example, in the operations of vehicles, robots, medical devices and home appliances, among others. At the higher end of microcontroller capability, the term system-on-a-chip (SoC) is often used, though there is no exact delineation in terms of RAM, clock speed and so on.
The embedded market was estimated to be more than $140 billion in 2013, with many analysts projecting a market larger than $20 billion by 2020.
Manufacturers of chips for embedded systems include many mainstays of the computer world, such as Apple, IBM, Intel, and Texas Instruments, but also numerous other companies that are less familiar to those outside the field. One highly influential vendor in this space has been ARM, which began as an outgrowth of Acorn, a U.K. maker of early PCs. The RISC-based architecture of the ARM chip, produced under license by other companies, has been widely used in mobile phones and PDAs and remains the most widely deployed SoC in the embedded world, with billions of units fielded.

Embedded Systems Software

A typical industrial microcontroller is quite unsophisticated compared to a typical enterprise desktop computer and generally depends on a simpler, less-memory-intensive program environment. The simplest devices run on bare

metal and are programmed directly using the chip CPU's machine code language.
Often, however, embedded systems use operating systems or language platforms tailored to Embedded use, particularly where real-time operating environments must be served. At higher levels of chip capability, such as those found in SoCs, designers have increasingly decided that the systems are generally fast enough and tasks tolerant of slight variations in reaction time that "near-real-time" approaches are suitable. In these instances, stripped-down versions of the Linux operating system are commonly deployed, though there are also other operating systems that have been pared down to run on embedded systems, including Embedded Java and Windows IoT (formerly Windows Embedded).
Generally, storage of programs and operating systems on embedded devices make use either of flash or rewritable flash memory.

Embedded Systems Classifications

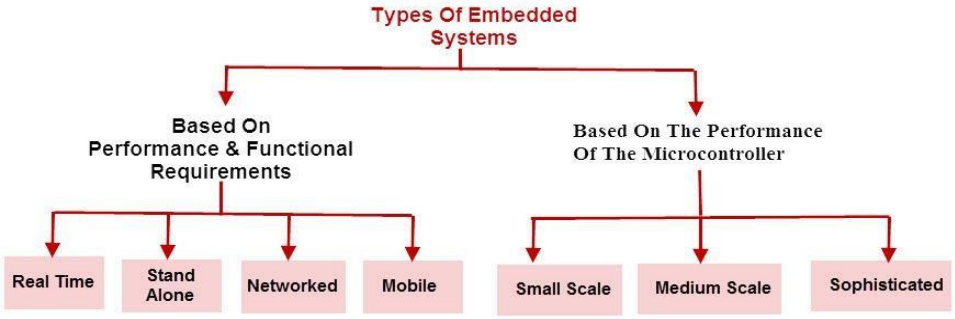


FIGURE 2: TYPES OF EMBEDDED SYSTEMS

Embedded systems are primarily classified into different types based on complexity of hardware & software and microcontroller (8 or 16 or 32-bit). Thus, based on the Performance of the microcontroller, embedded systems are classified into three types such as:

- Small scale embedded systems.
- Medium scale embedded systems.

- Sophisticated embedded systems.

Further, based on performance and functional requirements of the system embedded system classified into four types such as:

- Real time embedded systems.
- Stand-alone embedded systems.
- Networked embedded systems.
- Mobile embedded systems.

Real Time Operating System

A Real Time Operating System, commonly known as an RTOS, is a software component that rapidly switches between tasks, giving the impression that multiple programs are being executed at the same time on a single processing core.

In actual fact the processing core can only execute one program at any one time, and what the RTOS is actually doing is rapidly switching between individual programming threads (or Tasks) to give the impression that multiple programs are executing simultaneously.

Operating System or RTOS?

The difference between an OS (Operating System) such as Windows or Unix and an RTOS (Real Time Operating System) found in embedded systems, is the response time to external events. OS's typically provide a non-deterministic, soft real time response, where there are no guarantees as to when each task will complete, but they will try to stay responsive to the user. An RTOS differs in that it typically provides a hard-real time response, providing a fast, highly deterministic reaction to external events. The difference between the two can be highlighted through examples – compare, for example, the editing of a document on a PC to the operation of a precision motor control.

Types of RTOS

Hard Real Time:

In Hard RTOS, the deadline is handled very strictly which means that given task must start executing on specified scheduled time and must be completed within the assigned time duration.

Example: Medical critical care system, Aircraft systems, etc.

Firm Real time:

These type of RTOS also need to follow the deadlines. However, missing a deadline may not have big impact but could cause undesired affects, like a huge reduction in quality of a product.

Example: Various types of Multimedia applications.

Soft Real Time:

Soft Real time RTOS, accepts some delays by the Operating system. In this type of RTOS, there is a deadline assigned for a specific job, but a delay for a small amount of time is acceptable. So, deadlines are handled softly by this type of RTOS.

Example: Online Transaction system and Livestock price quotation System.

Scheduling Algorithms

When switching between Tasks the RTOS must choose the most appropriate task to load next. There are several scheduling algorithms available, including Round Robin, Co-operative and Hybrid scheduling. However, to provide a responsive system most RTOS's use a pre-emptive scheduling algorithm.

Task & Priorities

In a pre-emptive system each Task is given an individual priority value. The faster the required response, the higher the priority level assigned. When working in pre-emptive mode, the task chosen to execute is the highest priority task that can execute. This results in a highly responsive system.

Why RTOS?

There are well-established techniques for writing good, embedded software without the use of an RTOS. In some cases, these techniques may provide the most appropriate solution; however, as the solution becomes more complex, the benefits of an RTOS become more apparent. These include:

Priority Based Scheduling: The ability to separate critical processing from non-critical is a powerful tool.

Abstracting Timing Information: The RTOS is responsible for timing and provides API functions. This allows for cleaner (and smaller) application code.

Maintainability/Extensibility: Abstracting timing dependencies and task-based design results in fewer interdependencies between modules. This makes for easier maintenance.

Modularity: The task-based API naturally encourages modular development as a task will typically have a clearly defined role.

Promotes Team Development: The task-based system allows separate designers/teams to work independently on their parts of the project.

Easier Testing: Modular task-based development allows for modular task-based testing.

Code Reuse: Another benefit of modularity is that similar applications on similar platforms will inevitably lead to the development of a library of standard tasks.

Improved Efficiency: An RTOS can be entirely event driven; no processing time is wasted polling for events that have not occurred.

Idle Processing: Background or idle processing is performed in the idle task. This ensures that things such as CPU load measurement, background CRC checking etc. will not affect the main processing.

Applications of RTOS

Real-time systems are used in:

- Airlines reservation system.
- Air traffic control system.
- Systems that provide immediate updating.

- Used in any system that provides up to date and minute information on stock prices.
- Defense application systems like RADAR.
- Networked Multimedia Systems.
- Command Control Systems.
- Internet Telephony.
- Anti-lock Brake Systems.
- Heart Pacemaker.

Internet of Things (IoT)

Simply, this is the concept of basically connecting any device with an on and off switch to the Internet (and/or to each other). This includes everything from cellphones, coffee makers, washing machines, headphones, lamps, wearable devices and almost anything else you can think of. This also applies to components of machines, for example a jet engine of an airplane or the drill of an oil rig. As I mentioned, if it has an on and off switch then chances are it can be a part of the IoT. The analyst firm Gartner says that by 2020 there will be over 26 billion connected devices... That is a lot of connections (some even estimate this number to be much higher, over 100 billion). The IoT is a giant network of connected "things" (which also includes people). The relationship will be between people-people, people-things, and things-things.

IoT Architecture

There are four main components of an IOT system:

- The Thing itself (the device).
- The Local Network: this can include a gateway, which translates proprietary communication protocols to Internet Protocol.
- The Internet.
- Back-End Services; enterprise data systems, or PCs and mobile devices.

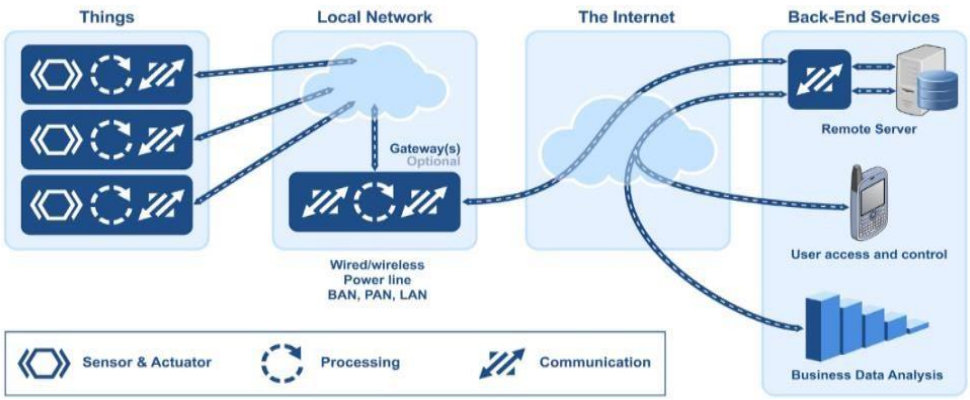


FIGURE 3: IOT ARCHITECTURE

Embedded Things

The definition of a "Thing" on the Internet of Things varies a lot, we define a Thing as an embedded computing device (or embedded system) that transmits and receives information over a network.

Embedded systems are based on microcontrollers (MCUs) and run software with a small memory footprint. Some Linux and Android-based systems can also be described as embedded systems. But usually, these general-purpose operating systems require an application processor, and have additional capabilities such as dynamic application loading. Therefore MCU-based embedded systems are often described as deeply embedded systems, versus the more general definition of embedded systems.

FIGURE 4: IOT DEVICES

Local Network

Your choice of communication technology directly affects your device's hardware requirements and costs. Which networking technology is the best choice? IoT devices are deployed in so many ways — in clothing, houses, buildings, campuses, factories, and even in your body — that no single networking technology can fit all bills.

Let us take a factory as a typical case for an IoT system. A factory would need many connected sensors and actuators scattered over a wide area, and a wireless technology would be the best fit.

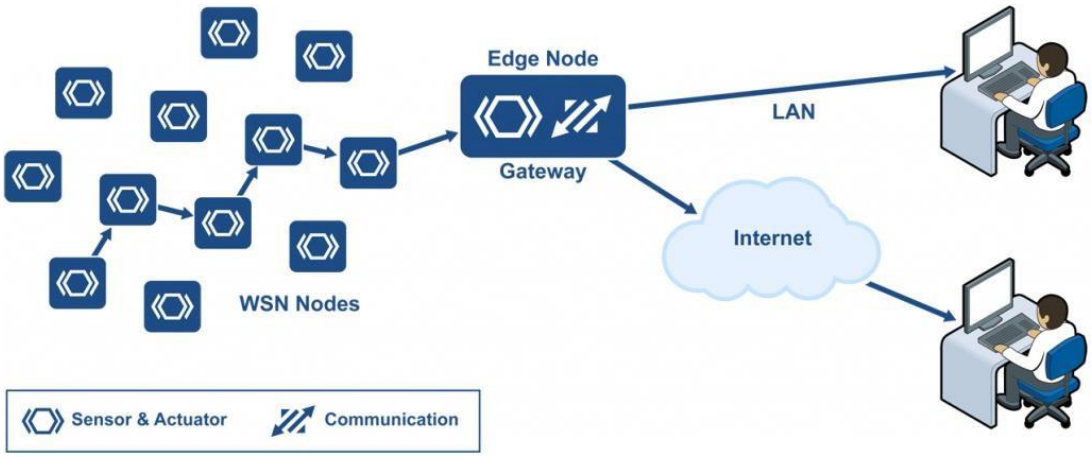


FIGURE 5: IOT NETWORK

A wireless sensor network (WSN) is a collection of distributed sensors that monitor physical or environmental conditions, such as temperature, sound, and pressure. Data from each sensor passes through the network node-to-node.

WSN Nodes

WSN nodes are low cost devices, so they can be deployed in high volume. They also operate at low power so that they can run on battery, or even use energy harvesting. A WSN node is an embedded system that typically performs a single function (such as measuring temperature or pressure or turning on a light or a motor).

Energy harvesting is a new technology that derives energy from external sources (for example, solar power, thermal energy, wind energy, electromagnetic radiation, kinetic energy, and more). The energy is captured

and stored for use by small, low-power wireless autonomous devices, like the nodes on a WSN.

WSN Edge Nodes

A WSN edge node is a WSN node that includes Internet Protocol connectivity. It acts as a gateway between the WSN and the IP network. It can also perform local processing, provide local storage, and can have a user interface.

WSN Technologies

The first obvious networking technology candidate for an IoT device is Wi-Fi, because it is so ubiquitous. Certainly, Wi-Fi can be a good solution for many applications. Almost every house that has an Internet connection has a Wi-Fi router.

However, Wi-Fi needs a fair amount of power. There are myriad devices that cannot afford that level of power: battery operated devices, for example, or sensors positioned in locations that are difficult to power from the grid.

IoT Protocols

HTTP:

It is the foundation of the client-server model used for the Web. The more secure method to implement HTTP is to include only a client in your IoT device, not a server. In other words, it is safer to build an IoT device that can only initiate connections, not receive. After all, you do not want to allow outside access to your local network.

MQTT:

MQ Telemetry Transport (MQTT) is an open source protocol for constrained devices and low-bandwidth, high-latency networks. It is publish/subscribe messaging transport that is extremely lightweight and ideal for connecting small devices to constrained networks.

MQTT is bandwidth efficient, data agnostic, and has continuous session awareness. It helps minimize the resource requirements for your IoT device,

while also attempting to ensure reliability and some degree of assurance of delivery with grades of service.
MQTT targets large networks of small devices that need to be monitored or controlled from a back-end server on the Internet. It is not designed for device-to-device transfer. Nor is it designed to "multicast" data to many receivers. MQTT is extremely simple, offering few control options.

Applications

Energy Management:

Integration of sensing and actuation systems, connected to the Internet, is likely to optimize energy consumption. It is expected that IoT devices will be integrated into all forms of energy consuming devices (switches, power outlets, bulbs, televisions, etc.) and be able to communicate with the utility supply company in order to effectively balance power generation and energy usage. Such devices would also offer the opportunity for users to remotely control their devices, or centrally manage them via a cloud based interface, and enable advanced functions like scheduling (e.g., remotely powering on or off heating systems, controlling ovens, changing lighting conditions etc.).

Medical Healthcare:

IoT devices can be used to enable remote health monitoring and emergency notification systems. These health monitoring devices can range from blood pressure and heart rate monitors to advanced devices capable of monitoring specialized implants, such as pacemakers Fitbit electronic wristbands or advanced hearing aids. Some hospitals have begun implementing "smart beds" that can detect when they are occupied and when a patient is attempting to get up. It can also adjust itself to ensure appropriate pressure and support is applied to the patient without the manual interaction of nurses. Specialized sensors can also be equipped within living spaces to monitor the health and general well- being of senior citizens.

Transportation:

The IoT can assist in integration of communications, control, and information processing across various transportation systems. Application of the IoT

extends to all aspects of transportation systems (i.e. the vehicle, the infrastructure, and the driver or user). Dynamic interaction between these components of a transport system enables inter and intra vehicular communication, smart traffic control, smart parking, electronic toll, collection systems, logistic and fleet management, vehicle control, and safety and road assistance.

IoT Challenges

Fragmentation:

IoT suffers from platform fragmentation and lack of technical standards a situation where the variety of IoT devices, in terms of both hardware variations and differences in the software running on them, makes the task of developing applications that work consistently between different inconsistent technology ecosystems hard. Customers may be hesitant to bet their IoT future on a proprietary software or hardware devices that uses proprietary protocols that may fade or become difficult to customize and interconnect.

Scalability:

According to a report put out by Gartner, 25 billion "things" will be connected to the internet by the year 2020. That's a pretty incredible estimation, considering the same report notes that 4.9 billion devices will be connected in 2015. This purported 400% increase in growth in only five years sheds some light on how much exponential IoT growth we can expect to see in the next 10, 20, or even 50 years.

Given these numbers, it is easy to understand why IPv6 (and its trillions upon trillions of new addresses) are important for IoT devices. Creators of IoT products that are connected over TCP/IP can rest assured that there will be a unique identifier available for their devices for a long, long time.

Data Storage and Analysis:

A challenge for producers of IoT applications is to clean, process and interpret the vast amount of data which is gathered by the sensors. There is a solution proposed for the analytics of the information referred to as Wireless Sensor

Networks. These networks share data among sensor nodes that are send to a distributed system for the analytics of the sensory data.
Another challenge is the storage of this bulk data. Depending on the application there could be high data acquisition requirements which in turn lead to high storage requirements. Currently the internet is already responsible for 5% of the total energy generated and this consumption will increase significantly when we start utilizing applications with multiple embedded sensors.

Security:

The fundamental security weakness of the Internet of Things is that it increases the number of devices behind your network's firewall. Ten years ago, most of us had to only worry about protecting our computers. Five years ago, we had to worry about protecting our smartphones as well. Now we must worry about protecting our car, our home appliances, our wearables, and many other IoT devices.
Because there are so many devices that can be hacked, that means that hackers can accomplish more. You may have heard about how hackers could potentially remotely control cars and remotely accelerate or decelerate the car. But hackers could use even seemingly unimportant devices like baby monitors or your thermostat to uncover private information or just ruin your day. The point is that we must think about what a hacker could do with a device if he can break through its security.

Updates:

As the Internet of Things becomes reality, we have to worry about protecting more devices. But even if you start taking security seriously, the tech companies which make these new devices are too cavalier about the risks. And one problem is that companies do not update their devices enough or at all. This means that an IoT device which was safe when you first bought it can become unsafe as hackers discover new vulnerabilities.
Computers used to have this problem, but automatic and easier updates have helped alleviate this problem. But as CSO points out, companies pressured to get their devices out quickly end up compromising on security. Even if they

may offer firmware upgrades for a time, they often stop when they focus on constructing the next device, leaving customers with slightly outdated hardware that can become a security risk.

Chapter 2

Introduction

In the project the system start remotely or using IoT and it is based on Real Time Operating System (RTOS) to detect the obstacles front the car during manually moving forward and stop the car immediately if an object detected, If the light on the light system helps the driver with automatic adjust due to the situation of the car and the environment, The mirrors system can control the mirrors position using it's keys to open, close, increase angle and decrease angle of each mirror, the automatic car parking system starting when the car next to the column of cars or spots then press the auto parking key and the car will find the spot if available then auto start the auto parallel parking mechanism, if the auto parking stop key pressed then the auto parking immediately stop and the driver can control the car manually.

Hardware Modules

This includes all the hardware used in the system.

Car Body

The car is equipped with two DC Motor s for transitional movement, a Steering Mechanism and Power Supply for electrical instruments.

Ultrasonic Sensor

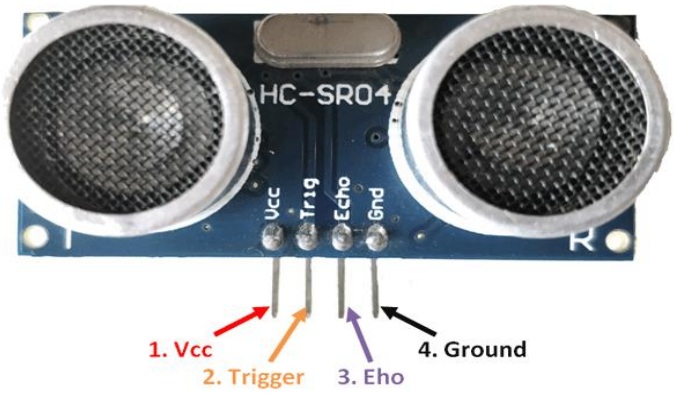


FIGURE 6: ULTRASONIC SENSOR

The ultrasonic sensor works on the principle of SONAR and RADAR system which is used to determine the distance to an object.
An ultrasonic sensor generates the high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver as shown in below figure.

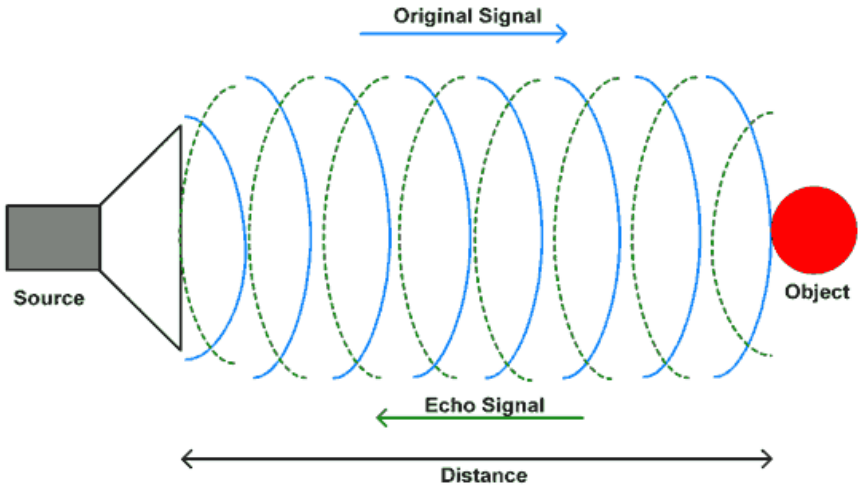


FIGURE 7: ULTRASONIC OPERATION

By measuring the time required for the echo to reach to the receiver, we can calculate the distance. This is the basic working principle of Ultrasonic module to measure distance.
HC-SR-04 has an ultrasonic transmitter, receiver, and control circuit.
In ultrasonic module HCSR04, we must give trigger pulse, so that it will generate ultrasound of frequency 40 kHz. After generating ultrasound i.e. 8 pulses of 40 kHz, it makes echo pin high. Echo pin remains high until it does not get the echo sound back. So, the width of echo pin will be the time for sound to travel to the object and return. Once we get the time, we can calculate distance, as we know the speed of sound.
HC-SR04 can measure up to range from 2 cm - 400 cm.

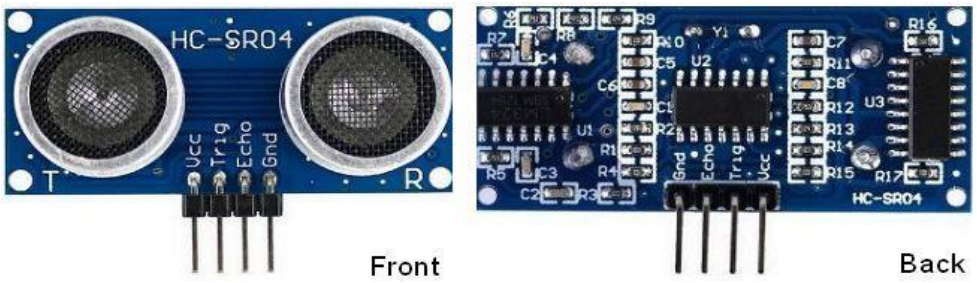


FIGURE 8: ULTRASONIC PIN MAPPING

- VCC – +5V Supply.
- TRIG – Trigger input of sensor. Microcontroller applies 10us trigger pulse to the HC-SR04 ultrasonic module.
- ECHO – Echo output of sensor. Microcontroller reads/monitors this pin to detect the obstacle or to find the distance.
- GND – Ground.

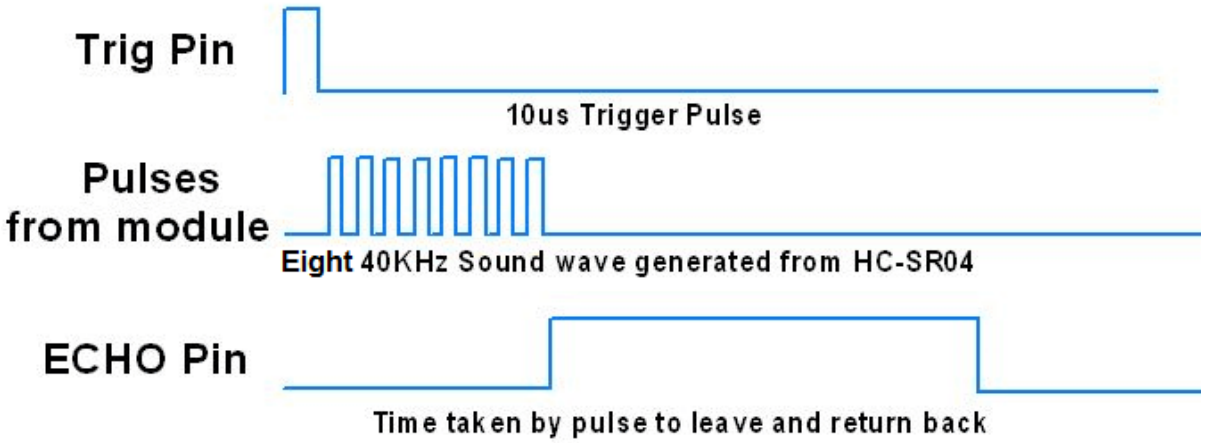


FIGURE 9: ULTRASONIC TIMING DIAGRAM

The operating sequence is as follows:

1. We need to transmit trigger pulse of at least 10 us to the HC-SR04 Trig Pin.
2. Then the HC-SR04 automatically sends Eight 40 kHz sound wave and wait for rising edge output at Echo pin.
3. When the rising edge capture occurs at Echo pin, start the Timer, and wait for falling edge on Echo pin.

4. As soon as the falling edge is captured at the Echo pin, read the count of the Timer. This time count is the time required by the sensor to detect an object and return from an object.

To get the distance, it is known that

$$Distance = Speed \times Time$$

The speed of sound is 343 m/s, therefore the distance between an object and the sensor

$$Distance = \frac{343 \times Time\ of\ Flight}{2}$$

LED Matrix

A LED matrix or LED display is a large, low-resolution form of dot-matrix display, useful both for industrial and commercial information displays as well as for hobbyist human–machine interfaces. It consists of a 2-D diode matrix with their cathodes joined in rows and their anodes joined in columns (or vice versa). By controlling the flow of electricity through each row and column pair it is possible to control each LED individually. By multiplexing, scanning across rows, quickly flashing the LEDs on and off, it is possible to create characters or pictures to display information to the user.[4] By varying the pulse rate per LED, the display can approximate levels of brightness. Multi-colored LEDs or RGB-colored LEDs permit use as a full-color image display. The refresh rate is typically fast enough to prevent the human eye from detecting the flicker.

The primary difference between a common LED matrix and an OLED display is the large, low resolution dots. The OLED monitor functionally works the same, except there are many times more dots, and they are all much smaller, allowing for greater detail in the displayed patterns.

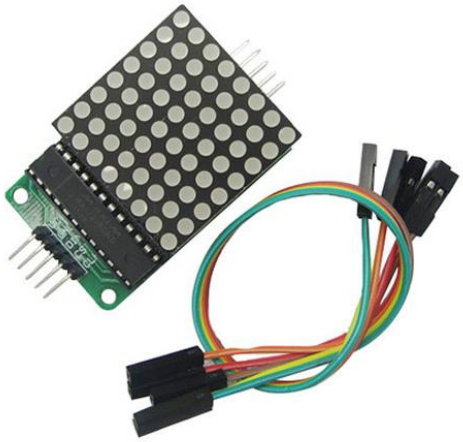


FIGURE 10: LED MATRIX

The LED Matrix above is the one used in the system, it consists of 8x8 LEDs connected, it operates at 5V. The Matrix comes with a module called Max7219 that works with SPI, to avoid using 16 pins.
The module has 8 registers, one for each column, through SPI the one byte is sent with the command with one bit for each LED. The module has repeater that keeps repeating the last command sent to it.

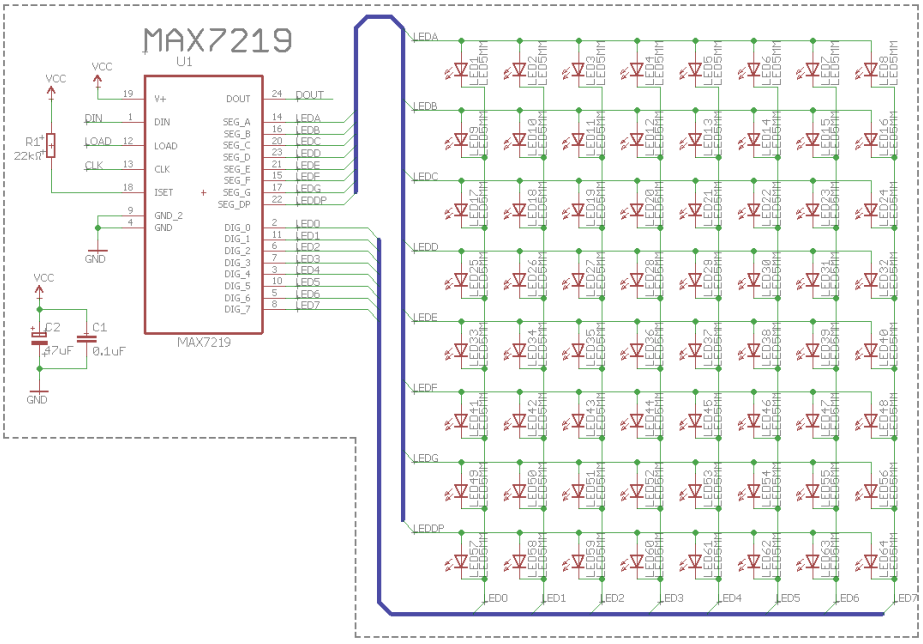


FIGURE 11: LED MATRIX SCHEMATIC

The schematic above is how the module and the matrix are connected together and to the microcontroller.

Stepper Motor

A stepper motor, also known as step motor or stepping motor, is a brushless DC electric motor that divides a full rotation into several equal steps. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed. Switched reluctance motors are very large stepping motors with a reduced pole count, and generally are closed loop commutated.

Brushed DC motors rotate continuously when DC voltage is applied to their terminals. The stepper motor is known by its property of converting a train of input pulses (typically square wave pulses) into a precisely defined increment in the shaft position. Each pulse moves the shaft through a fixed angle.

Stepper motors effectively have multiple "toothed" electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external driver circuit or a micro controller. To make the motor shaft turn, first, one electromagnet is given power, which magnetically attracts the gear's teeth. When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. This means that when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one. From there the process is repeated. Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.

The circular arrangement of electromagnets is divided into groups, each group called a phase, and there is an equal number of electromagnets per group. The number of groups is chosen by the designer of the stepper motor. The electromagnets of each group are interleaved with the electromagnets of other groups to form a uniform pattern of arrangement. For example, if the stepper motor has two groups identified as A or B, and ten electromagnets in total, then the grouping pattern would be ABABABABAB.

Electromagnets within the same group are all energized together. Because of this, stepper motors with more phases typically have more wires (or leads) to control the motor.

There are many types of Stepper Motors: Unipolar, Bipolar, etc. The motors we used are Unipolar Steppers.

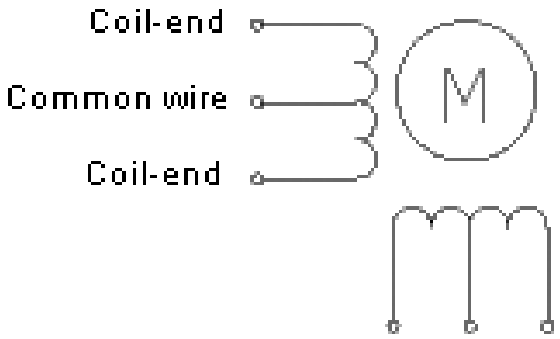


FIGURE 12: STEPPER MOTOR

A unipolar stepper motor has one winding with center tap per phase. Each section of windings is switched on for each direction of magnetic field. Since in this arrangement a magnetic pole can be reversed without switching the direction of current, the commutation circuit can be made very simple (e.g., a single transistor) for each winding. Typically, given a phase, the center tap of each winding is made common: giving three leads per phase and six leads for a typical two-phase motor. Often, these two-phase commons are internally joined, so the motor has only five leads.

A microcontroller or stepper motor controller can be used to activate the drive transistors in the right order, and this ease of operation makes unipolar motors popular with hobbyists; they are probably the cheapest way to get precise angular movements. For the experimenter, the windings can be identified by touching the terminal wires together in PM motors. If the terminals of a coil are connected, the shaft becomes harder to turn. One way to distinguish the center tap (common wire) from a coil-end wire is by measuring the resistance. Resistance between common wire and coil-end wire is always half of the resistance between coil-end wires. This is because there is twice the length of coil between the ends and only half from center (common wire) to the end. A quick way to determine if the stepper motor is working is to short circuit every two pairs and try turning the shaft. Whenever a higher than normal resistance is felt, it indicates that the circuit to the winding is closed and that the phase is working.

The stepper motor requires higher current supply so, A Darlington pair is used. In electronics, a multi-transistor configuration called the Darlington configuration (commonly called a Darlington pair) is a compound structure of a particular design made by two bipolar transistors connected in such a way that the current amplified by the first transistor is amplified further by the second one. This configuration gives a much higher current gain than each transistor taken separately. It was invented in 1953 by Sidney Darlington. It's as follows.

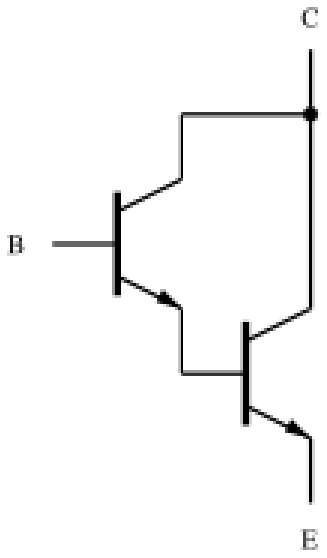


FIGURE 13: STEPPER MOTOR - DARLINGTON PAIR

The connection to the microcontroller is as follows

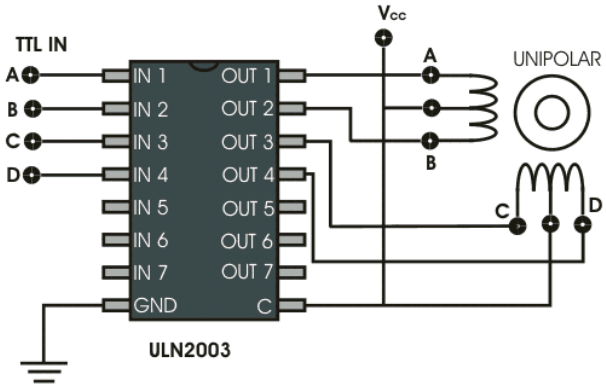


FIGURE 14: STEPPER MOTOR & DRIVER PIN MAPPING

Bluetooth Module

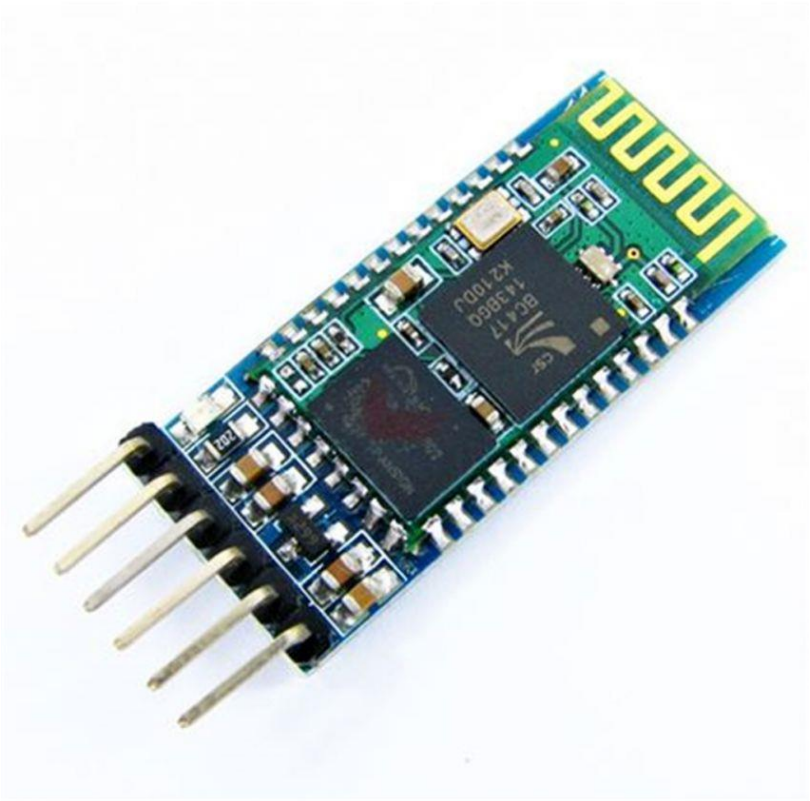


FIGURE 15: BLUETOOTH MODULE

The Bluetooth module HC-05 is a MASTER/SLAVE module. By default, the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc.

Hardware Features

- Typical -80dBm sensitivity.
- Up to +4dBm RF transmit power.
- 3.3 to 5 V I/O.
- PIO (Programmable Input/Output) control.
- UART interface with programmable baud rate.
- With integrated antenna.

- With edge connector.

Software Features

- Slave default Baud rate: 9600, Data bits:8, Stop bit:1, Parity: No parity.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"1234" as default.

Module pin mapping

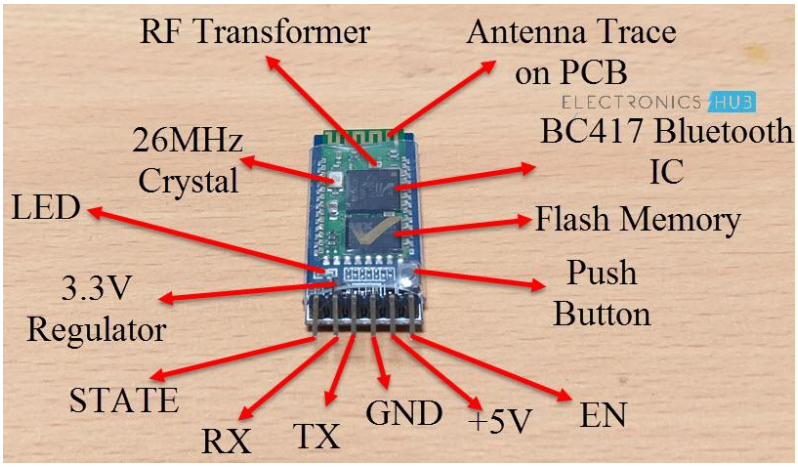


FIGURE 16: BLUETOOTH MODULE PIN MAPPING

Wi-Fi Module

FIGURE 17: WI-FI MODULE

The ESP8266 is a low-cost Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability, produced by Espressif Systems in Shanghai, China. The chip first came to the attention of Western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at first there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, the chip, and the software on it, as well as to translate the Chinese documentation.

The ESP8285 is an ESP8266 with 1 MiB of built-in flash, allowing the building of single-chip devices capable of connecting to Wi-Fi.

The pinout is as follows for the common ESP-01 module:

- VCC, Voltage (+3.3 V; can handle up to 3.6 V)
- GND, Ground (0 V)
- RX, Receive data bit X
- TX, Transmit data bit X
- CH_PD, Chip power-down
- RST, Reset
- GPIO 0, General-purpose input/output No. 0
- GPIO 2, General-purpose input/output No. 2

Software Components

The system is RTOS based and consist of six tasks each task use number of software modules and drivers to obtain its goals.

#T_CarControl

Task type	Used modules and drivers	Modules and drivers APIs	Notes

On Event task	Car_control	void car_control_init(void);
		void move_forward(void);
		void move_backward(void);
		void move_right(void);
		void move_left(void);
		void car_stop(void);
		void car_speed_change(void);	

	bit_handle	setBit(R,N) clearBit(R,N)	

#T_BlueTooth

Task type	Used modules and drivers	Modules and drivers APIs	Notes

Periodic task	UART	void Uart_Init(UART_Type uartNum, INT32U u32Baud);
		void Uart_SendByte(UART_Type uartNum, INT8U u8Data);
		void Uart_SendStr(UART_Type uartNum, INT8U* pu8Str);
		INT8U Uart_ReceiveByte(UART_Type uartNum);
		Bool Uart_ReceiveByte_Unblock(UART_Type uartNum, INT8U* pu8Data);
		void Uart0_Rec_Int_Enable(void);
		void Uart1_Rec_Int_Enable(void);
		void Uart0_Rec_Int_Disable(void);
		void Uart1_Rec_Int_Disable(void);
		INT8U Uart0_ImmedReceiveByte(void);
		INT8U Uart1_ImmedReceiveByte(void);	

#T_Light

Task type	Used modules and drivers	Modules and drivers APIs	Notes

On Event task	LEDMatrix	void LEDMatrix_Init(void);
		void LEDMatrix_AllOn(void);
		void LEDMatrix_AllOff(void);
		void LEDMatrix_AllDim(void);
		void LEDMatrix_MidOff(void);
		void LEDMatrix_RightHalf(void);
		void LEDMatrix_LeftHalf(void);
			This module uses only the SPI peripherals and its pins
			SPI_PORT PORTB
			SPI_DDR DDRB
			
			SPI_SS PB0
			SPI_SCK PB1
			SPI_MOSI PB2

	SPI	void SPI_Init(SPI_Mode spiMode); INT8U SPI_Tranceive(INT8U u8Data);	

#T_Mirrors.

Task type	Used modules and drivers	Modules and drivers APIs	Notes

On Event task	Unipolar_stepper	void uni_stepper_init(INT8U);
		void uni_stepper_increment_clkwise(INT8U);
		void uni_stepper_increment_anticlk(INT8U);	

#T_ObjectDetection

Task type	Used modules and drivers	Modules and drivers APIs	Notes

Periodic task	Ultrasonic	void ULTRASONIC_Init(void);
		void ULTRASONIC_Start(ULTRASONIC_type ultrasonic_pin);
		INT16U ULTRASONIC_GetDistanceNoBlock(INT16U*Pdistance);	

	Timer3	void Timer3_Init(Timer3Mode_type mode,Timer3Scaler_type scaler,OC3A_Mode_type oc3a_mode,OC3B_Mode_type oc3b_mode);
		void Timer3_InputCaptureEdge(ICU_Edge_type edge);
		void Timer3_ICU_InterruptEnable(void);
		void Timer3_ICU_InterruptDisable(void);
		void Timer3_OVF_InterruptEnable(void);
		void Timer3_OVF_InterruptDisable(void);	

#T_Parking

Task type	Used modules and drivers	Modules and drivers APIs	Notes

On Event task	Car Control	void car_control_init(void);
		void move_forward(void);
		void move_backward(void);
		void move_right(void);
		void move_left(void);
		void car_stop(void);
		void car_speed_change(void);	

	Ultrasonic	void ULTRASONIC_Init(void);
		void ULTRASONIC_Start(ULTRASONIC_type ultrasonic_pin);
		INT16U ULTRASONIC_GetDistanceNoBlock(INT16U*Pdistance);	

	Timer3	void Timer3_Init(Timer3Mode_type mode,Timer3Scaler_type scaler,OC3A_Mode_type oc3a_mode,OC3B_Mode_type oc3b_mode);
		void Timer3_InputCaptureEdge(ICU_Edge_type edge);
		void Timer3_ICU_InterruptEnable(void);	

		void Timer3_ICU_InterruptDisable(void);
		void Timer3_OVF_InterruptEnable(void);
		void Timer3_OVF_InterruptDisable(void);	

Chapter 3

Introduction

In this project we deliver a Driver Assistance System, that helps the driver while driving. This system consists of sub features.

1. Turning on the system happens through two ways, either you are the owner and therefore you can unlock the car with the Bluetooth connection from the mobile by entering a password. Another way if you are lending your car to someone else you can unlock the car for him and start the system using Wi-Fi.
2. Mirror Control, there are 6 commands that can be sent to the controller through Bluetooth, the commands open the mirrors completely, fold it completely, open or close with one step. A stepper motor was used for this application with 5.625 degrees step.
3. Adaptive light control, LED matrix changes its shape (which LEDs on or off) according to the distance between you and the car in front also according to the direction.
4. Obstacle Detection, this is done using ultrasonic sensors in front of the car to detect any object that jumps in the way and brakes the vehicle according to the distance between the vehicle and the object.
5. Auto Parking this is done by 2 Ultrasonics one is used to detect any object in front to delay the parking sequence and the other is used to measure the depth needed for the spot.

Bluetooth and Wi-Fi Control

In some situations some one other the car owner will need to use the car, for more secure and flexible user experience the Remote Authorization System allows the car owner to start his car over IOT.

The login to the car system offers to level of authorization:

First it asks the user if he is the owner of the car or he needs to ask the owner for the system login.

If he is the owner, it asks him for the password and stats the system.

If the user is not the owner, the system will start the WIFI module and the connection with the server waiting for the password form the owner to start the system.
After the driver legitimately access the car system, he can control the car over the Bluetooth.
The Bluetooth receives the driver commands and trigger the suitable function, controlling the car direction, the light on/off , the car speed , the mirrors and starting/stopping the auto-parking mode which will find the suitable slot according to the cat size start the parking sequence and stop the car and return the normal operation mode.

Mirror Control

The Mirror Control is a one of the tasks in the RTOS developed for the Controlling both mirrors. Each mirror has a stepper motor connected to a Darlington Pair Module, and each module is connected to a 4 GPIO Pins on the microcontroller. The task is based on a driver developed by one of our team members just for the steppers, the main functions used from the driver are incrementing one step clockwise and incrementing one step anti clockwise.
The step of the steppers used is 5.625 degrees. The main task waits for 6 commands from a message queue, the commands are sent to the task through Bluetooth task. The commands are as follows:

- 'O' this command turns both steppers to 90 degrees and the other 270 (-90) degrees.
- 'C' this command returns the stepper to the original position 0 degrees.
- 'P' this command increments the right stepper just one step clockwise.
- 'p' this command increments the right stepper just one step anti-clockwise.
- 'Q' this command increments the left stepper one step anti-clockwise.
- 'q' this command increments the left stepper one step clockwise.

These commands are protected by limiters, one for each stepper so the stepper will not add more steps above 90 degrees or reduce steps below 90 degrees.

There were also two functions developed in this task to make both mirrors open and close together completely not a single one at a time to make them synchronized.
The steppers were connected to the micro controller as the following schematic below:

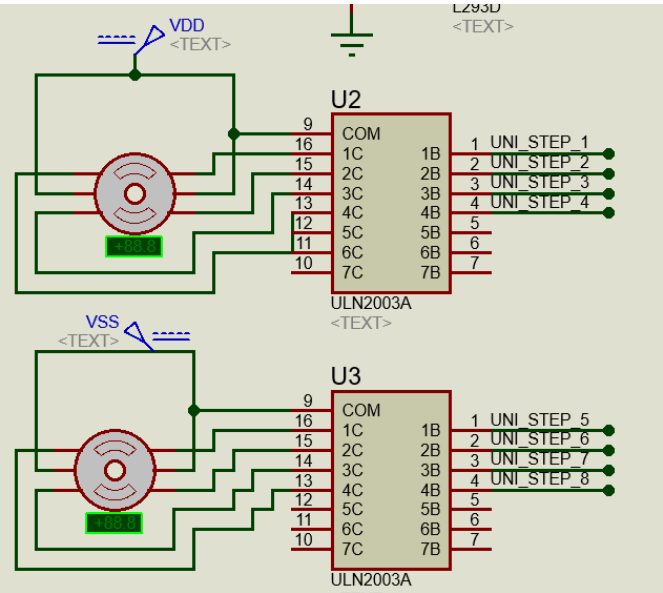


FIGURE 18: STEPPER MOTOR CONNECTION TO MICROCONTROLLER

Adaptive Light Control

Its role is to change the lighting according to the directions which the car is moving towards, and according to the other vehicles around the car. There are two main tasks the Adaptive Light Controls does, which are:

1. **Lighting only needed areas while cornering:**
   When the car turns to the right, it dims the light in the direction of the left area, and it lights all the right area. The same sequence happens when turning left. And this is done to help driver focus on areas he or she need to see clearly, moreover it helps in saving the power.

FIGURE 19: ADAPTIVE LIGHT CONTROL WHILE TURNING

2. **Avoiding over lighting areas in the rear view of other drivers.**
   Whenever there is a vehicle moving in front of the car, it turns of the area that could blind the other driver. And this is done for collaboration for safe driving.

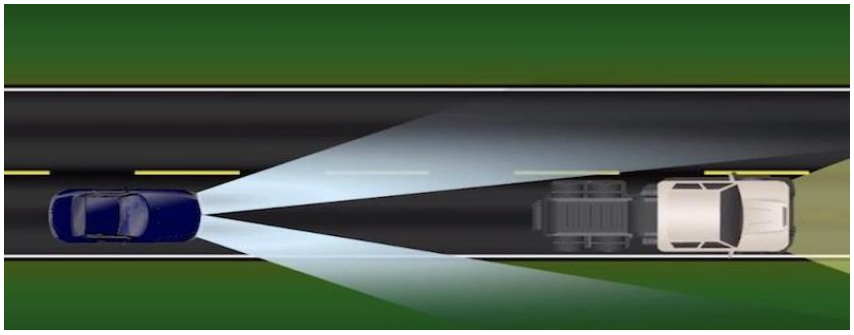


FIGURE 20: ADAPTIVE LIGHT CONTROL PREVENTING HEADLIGHT GLARE

LED Matrix Hardware Connection:

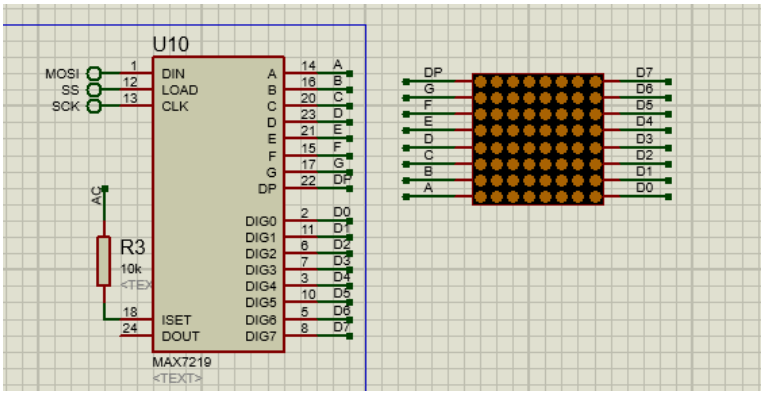


FIGURE 21: LEDMATRIX & MAX7219 MODULE CONNECTION TO MICROCONTROLLER

Obstacle Detection

By using Ultrasonic in the front of the car, that detect the Obstacles, we have three ranges:

- Less than 60 cm, in this case ultrasonic will send to task of car control there are object to stop the car and send to task of lights to make all lights on.
- Less than 120 cm and above 60 cm, in this case ultrasonic will send to task of car control there are no object near and send to task of lights to make middle lights off.

Above 120 cm, in this case ultrasonic will send to task of car control, task of lights there are clear and no objects and make all light.

The schematic of how the ultrasonic is connected to the micro controller, a diode was added on the echo pin to avoid signal alteration as we're using more than one Ultrasonic sensor.

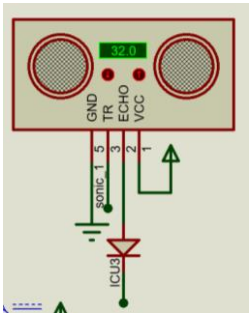


FIGURE 22: OBSTACLE AVOIDANCE ULTRASONIC SENSOR CONNECTION TO MICROCONTROLLER

Auto-Parking

To start the auto parallel parking algorithm the car must be next to the column of the cars or spots then send Parking command to the system through the Bluetooth then the scheduler will suspend all tasks and start the T_Parking to start searching for the spot this step use the front ultrasonic to detect if an obstacle found then stop the car till the way be clear and continue for searching for a spot, the right middle ultrasonic start reading the distance and if it less than the car width its continue searching for empty spot then if the ultrasonic read more than the car width and continue reading this value for time which indicates that is the spot is more than the car length and the car get a clear spot for parking.

After get the spot the car start the parking algorithm and it's a time based algorithm and using ultrasonic sensor for obstacles detection, then after the parking algorithm done the driver can control the car but during the auto parking the driver can't control the car before send Stop parking command to the system through the Bluetooth.

Static Design

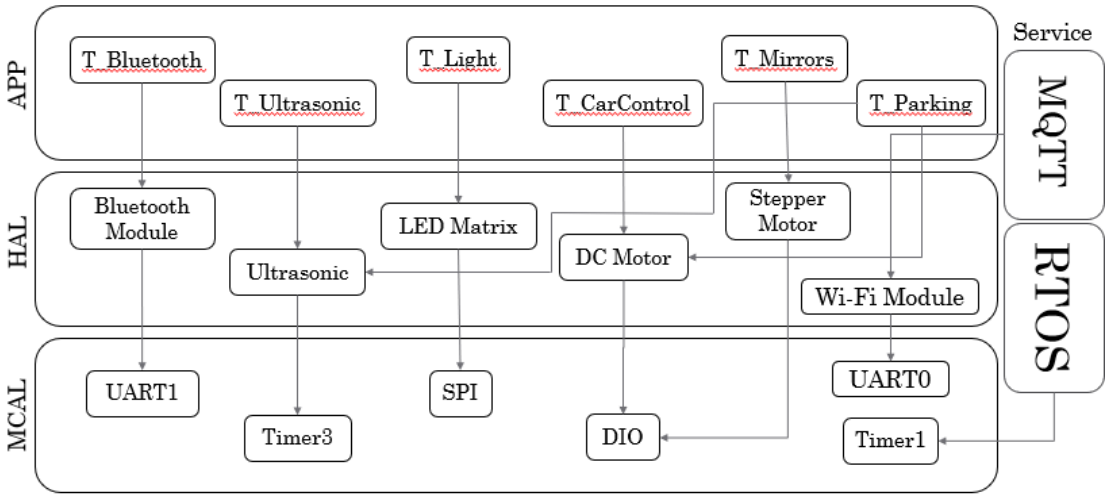


FIGURE 23: DAS STATIC DESIGN DIAGRAM

System Task Design

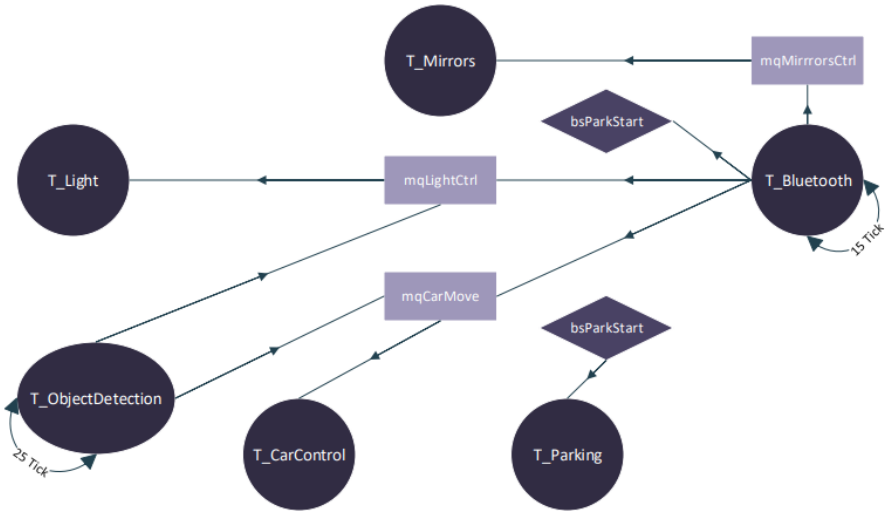


FIGURE 24: DAS TASK DESIGN DIAGRAM

Project Block Diagram

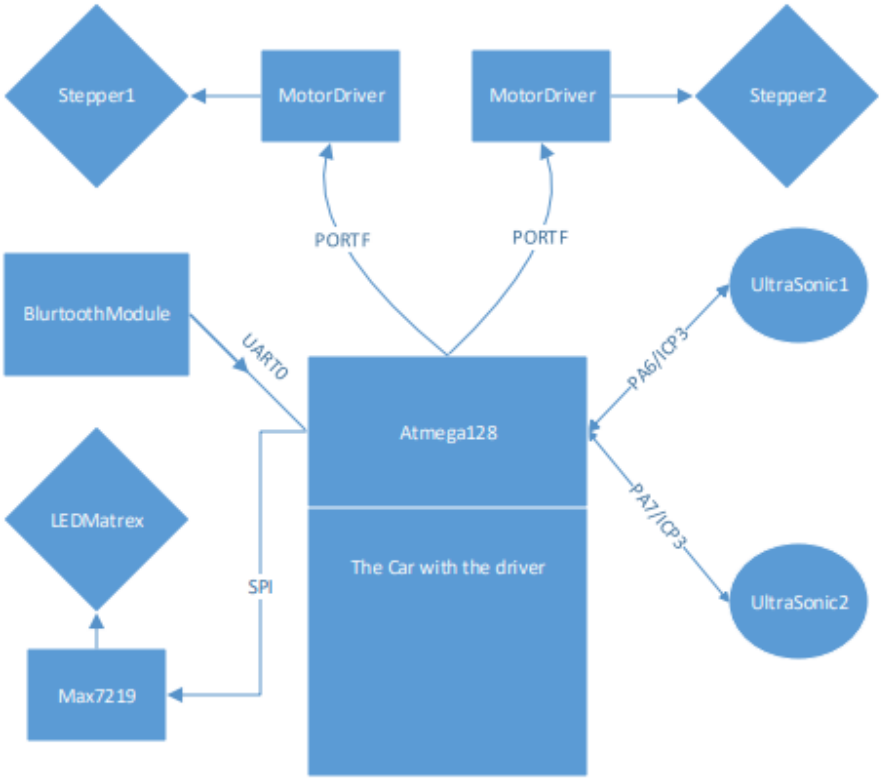


FIGURE 25: DAS BLOCK DIAGRAM

Testing Reports

Each Team Member had to test three other tasks made by the different teammate and the result of the testing reports are as following.

Task Name	T_Bluetooth
Test Type	Black Box / White Box
Reviewer Name	Ahmed Hossam
Code Author	Abdallah Ragab
Number of Issues/Bugs	0
Date	11/8/2020
Time Spent in minutes	10 minutes

Issues/ Bugs

Issue No.	**Description** (only the issue not the solution)	**Status**, filled by author Ex. Fixed, cannot fix, not a bug.
1		
2		
3		
4		

Task Name	T_Ultrasonic
Test Type	Black Box / White Box
Reviewer Name	Ahmed Hossam
Code Author	Mohamed Abdelkader
Number of Issues/Bugs	0
Date	11/8/2020
Time Spent in minutes	5

Issues/ Bugs

Issue No.	**Description** (only the issue not the solution)	**Status**, filled by author Ex. Fixed, cannot fix, not a bug.
1		
2		
3		
4		

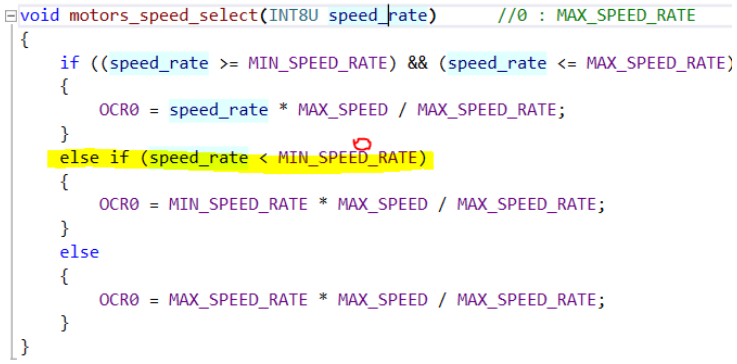
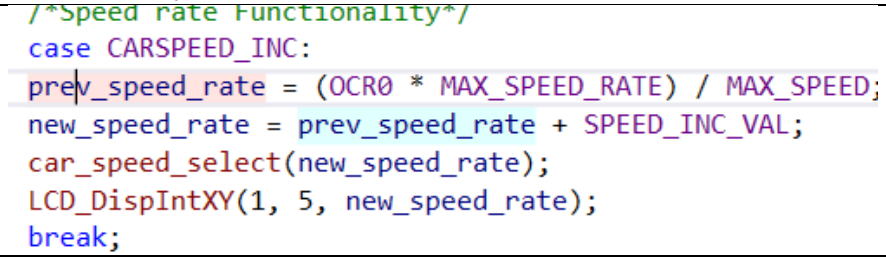
Task Name	T_Mirrors
Test Type	Black Box / White Box
Reviewer Name	Moustafa Raafat
Code Author	Ahmed Hossam
Number of Issues/Bugs	3
Date	11/8/2020
Time Spent in minutes	45

Issues/ Bugs

Issue No.	**Description** (only the issue not the solution)	**Status**, filled by author Ex. Fixed, cannot fix, not a bug.
1	Cases should be with #define values, not constants. Ex: use `MIR_OPEN` not 'O'	Fixed.
2	Suggestion: `UNI_360_STEPS` could be redefined directly as an integer (= 64 steps) value, to save time and space, as we don't have to use floating points.	Fixed.
3	Suggestion: You may consider writing a separate driver for the mirrors when having the hardware.	

Task Name	T_CarControl
Test Type	Black Box / White Box
Reviewer Name	Moustafa Raafat
Code Author	Hossam Mohamed
Number of Issues/Bugs	2
Date	12/8/2020
Time Spent in minutes	40

Issues/ Bugs

Issue No.	**Description** (only the issue not the solution)	**Status**, filled by author Ex. Fixed, can't fix, not a bug.
1	 This case will never happen, as INT8U cannot be < 0, so it's better to move both conditions to the task.c file (you've already done that with the < 0 condition).	
2		

	No need for the prev_speed variable (as well as all of the first line). We already can keep track of the speed with only one variable, as we know we start with speed_rate = 0, increasing could be Speed_ rate += SPEED_INC_VAL; Same with DEC: Speed_ rate -= SPEED_DEC_VAL;	
3		
4		

Task Name	T_UltraSonic
Test Type	Black Box / White Box
Reviewer Name	Moustafa Raafat
Code Author	Mohamed Abd ElKader
Number of Issues/Bugs	2
Date	12/8/2020
Time Spent in minutes	30

Issues/ Bugs

Issue No.	**Description** (only the issue not the solution)	**Status**, filled by author Ex. Fixed, cannot fix, not a bug.
1	`distance` Type should be changed to INT16U in T_Ultrasonic and in the Ultrasonic driver. As distance could be > 255 (cm); such a case will lead to an overflow!	

2	Suggestion: `Change_Flag` could be changed to an enum for readability. For example:
	`typedef enum{`
	`OBJ_NONE_STATE, OBJ_DTCTD, OBJ_NEAR, OBJ_CLR`
	`}ObjectPrevStatus_t;`
	As
	`OBJ_NONE_STATE` corresponds to 0xFF,
	`OBJ_NEAR` corresponds to 0,
	`OBJ_DTCTD` corresponds to 1,
	`OBJ_CLR` corresponds to 2,
	In the previous code.	

Task Name	T_Light
Test Type	Black Box / White Box
Reviewer Name	Mohamed Abd-el-Kader Mostafa
Code Author	Mostafa Raafat
Number of Issues/Bugs	0
Date	11/8/2020
Time Spent in minutes	10

Issues/ Bugs

Issue No.	**Description** (only the issue not the solution)	**Status**, filled by author Ex. Fixed, cannot fix, not a bug.
1		
2		
3		
4		

Task Name	T_Mirrors
Test Type	Black Box / White Box
Reviewer Name	Mohamed Abd-el-Kader Mostafa
Code Author	Ahmed Hossam
Number of Issues/Bugs	0
Date	11/8/2020
Time Spent in minutes	4 minutes

Issues/ Bugs

Issue No.	**Description** (only the issue not the solution)	**Status**, filled by author Ex. Fixed, cannot fix, not a bug.
1	For Only Open and Close Mirrors No Issues	
2		
3		
4		

Conclusion

This system tries to help the driver as much as possible, to make the driving experience much easier and better. This is done by the easy availability of unlocking the car and starting the system, mirror adjusting, light control, obstacle detection and lastly auto parking. However, there's still some room for improvements as we can use better sensors like LiDAR instead of Ultrasonics better readings and a wider range of data, GSM instead of Wi-Fi as Wi-Fi signals are weaker and not available in most public place in Egypt.