SMART TRAFFIC MANAGEMENT SYSTEM

A PROJECT REPORT

submitted by

Sai Khushee
230701276
Shankaranarayanan
230701306
Senthamizh selvi
230701301

in partial fulfillment for the award of the

degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this project **"Smart Traffic Management System"** is the bonafide work of "*Sai Khushee(230701276), Shankaranarayanan(230701306), Sentamizh(230701301)",* who carried out the project work under my supervision.

Faculty Incharge

Faculty name: Ponmani S

Designation: Assistant Professor

Department of Computer Science and Engineering, Rajalakshmi Engineering College, Chennai

This mini project report is submitted for the viva vice examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS:

1. ABSTRACT:

With increasing vehicle numbers and urban congestion, managing traffic effectively has become a major challenge. Traditional traffic light systems operate on fixed timers and do not adapt to varying traffic densities, leading to inefficiencies and longer wait times. This project proposes a smart traffic control system using IR sensors and RFID technology to optimize traffic flow and prioritize emergency vehicles.

Each lane is equipped with two IR sensors to monitor traffic density. When both sensors detect vehicle presence, it indicates a higher vehicle count, and the green light duration is extended for that lane. This ensures that heavily congested lanes receive more time, thereby reducing traffic buildup. Moreover, the system includes an RFID reader to detect emergency vehicles. When an emergency vehicle is near and its RFID tag is read, the traffic light instantly turns green for that lane, allowing quick and safe passage.

This automated system not only reduces congestion and wait times but also ensures emergency services are not delayed. By integrating low-cost sensors and microcontrollers, the proposed solution is affordable and scalable for smart city implementations.

2. OBJECTIVE:

The objective of this project is to design and implement an intelligent traffic light management system that dynamically adjusts the lane opening time based on real-time traffic density using IR sensors. Each lane is equipped with two IR sensors that count the number of vehicles, and if both sensors detect vehicle presence, the lane's green signal duration is increased. Additionally, an RFID-based mechanism is

integrated to prioritize emergency vehicles such as ambulances, fire engines, and police cars. When an emergency vehicle with an RFID tag approaches, the system identifies it and automatically opens the lane to ensure uninterrupted passage.

3. INTRODUCTION:

Urban areas are facing increasing challenges in managing traffic due to the rapid rise in vehicle usage. Traditional traffic lights operate on preset timers that fail to adapt to real-time traffic conditions, leading to inefficient traffic flow and frustration among commuters. Delays caused by traffic congestion also pose serious problems for emergency services that require swift navigation through busy streets.

A smart traffic management system is the need of the hour to reduce travel time, improve fuel efficiency, and enhance road safety. Modern technology, including embedded systems, IR sensors, and RFID, provides an opportunity to revolutionize how traffic is controlled. By using IR sensors placed at strategic points in each lane, we can measure vehicle presence and estimate traffic density. This data is then used to control the duration of the traffic signals dynamically.

Emergency response time is crucial in saving lives. Emergency vehicles often get stuck in traffic, despite using sirens. By integrating an RFID reader, emergency vehicles with an RFID tag can be detected, prompting the system to automatically grant them passage. This intelligent solution not only saves lives but also increases the efficiency of the entire traffic ecosystem.

The integration of IR sensors for density monitoring and RFID for emergency access provides a holistic approach to modern traffic

management, making it a step forward in building smart cities.

4. EXISTING SYSTEM:

In the existing traffic management systems, traffic lights operate on a fixed timing mechanism. The signal changes are based on predetermined time intervals regardless of the actual number of vehicles waiting at each lane. Traffic police may manually control the lights during peak hours or special situations, but this process is inefficient, labor-intensive, and prone to human error.

Additionally, emergency vehicles often struggle to navigate through traffic as there is no automated mechanism to detect their presence and adjust traffic signals accordingly. Delays in emergency services can result in severe consequences, especially in medical emergencies, fire accidents, and police interventions.

5. PROPOSED SYSTEM:

The proposed system utilizes two IR sensors per lane to estimate vehicle density. If both IR sensors in a lane detect vehicles, it indicates high traffic, prompting the microcontroller to increase the green light duration for that specific lane. This allows more vehicles to pass through and reduces congestion effectively.

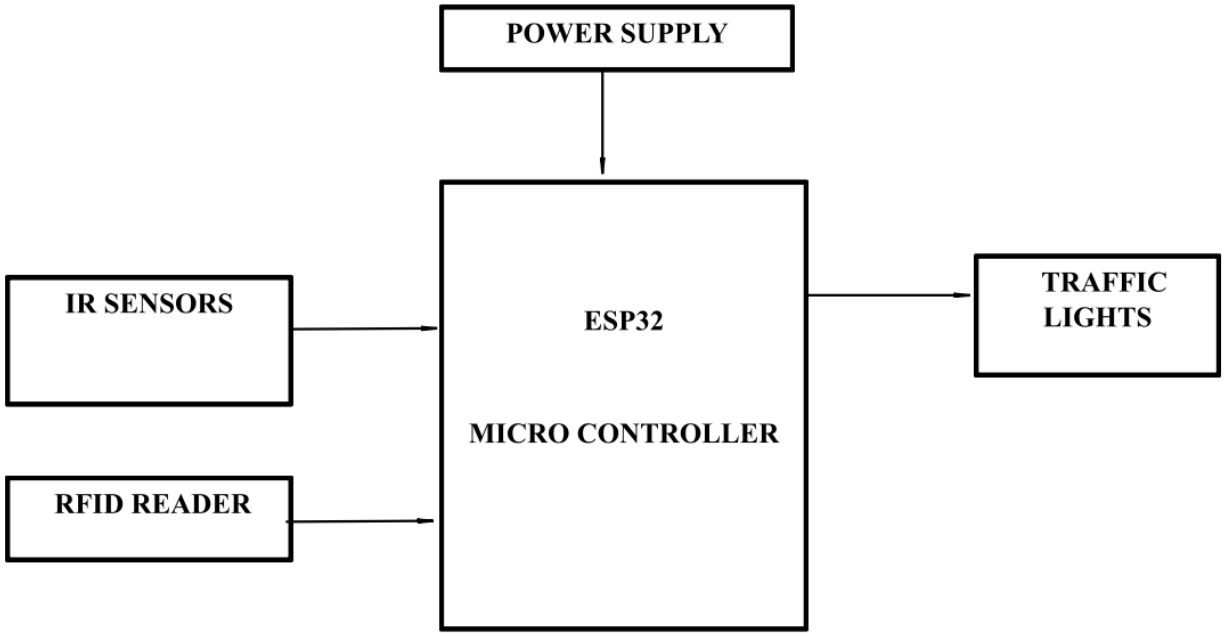
To handle emergency vehicles, each is fitted with an RFID tag. An RFID reader installed near the intersection detects the tag as the vehicle approaches. Upon recognizing a valid emergency RFID, the traffic light for that lane immediately turns green, regardless of its previous state. This ensures emergency vehicles get the right of way without delay.

The system is controlled using a microcontroller (like ESP32 or Arduino), and the logic is coded to manage the traffic lights based on sensor input. The entire process is automatic and requires minimal human intervention, leading to efficient and intelligent traffic control.

6. ADVANTAGES:

- **Real-time traffic detection** using IR sensors enhances efficiency.
- **Dynamic signal timing** improves vehicle flow and reduces waiting time.
- **Priority access for emergency vehicles** through RFID ensures public safety.
- **Minimizes human intervention**, reducing the chance of error.
- **Energy-efficient and cost-effective** system implementation.
- **Scalable and adaptable** for future smart city applications.
- **Reduces fuel consumption** and air pollution due to less idle time.
- **Improves emergency response time**, potentially saving lives.

7. BLOCK DIAGRAM:



8. REQUIREMENTS:

HARDWARE REQUIREMENTS

- ESP32 MICROCONTROLLER
- POWER SUPPLY
- IR SENSOR
- RFID READER
- TRAFFIC LIGHT

SOFTWARE REQUIREMENTS

- EMBEDDED C
- ARDUINO IDE

9. HARDWARE DESCRIPTION:

9.1. POWER SUPPLY ADAPTER

An adapter is a device that converts attributes of one electrical device or system to those of an otherwise incompatible device or system. Some modify power or signal attributes, while others merely adapt the physical form of one electrical connector to another. In a computer, an adapter is often built into a card that can be inserted into a slot on the computer's motherboard. The card adapts information that is exchanged between the computer's microprocessor and the devices that the card supports.

PRODUCT DESCRIPTION

An electric power adapter may enable connection of a power plug, sometimes called, used in one region to a AC power socket used in another, by offering connections for the disparate contact arrangements, while not changing the voltage. An AC adapter, also called a "recharger", is a small power supply that changes household electric current from distribution voltage) to low voltage DC suitable for consumer electronics. Some modify power or signal attributes, while others merely adapt the physical form of one electrical connector to another. For computers and related items, one kind of serial port adapter enables connections between 25-contact and nine-contact connectors, but does not affect electrical power- and signalling-related attributes.

FEATURES:

· Output current:1A

· Supply voltage: 220-230VAC

· Output voltage: 12VDC

· Reduced costs

· Increased value across front-office and back-office functions

· Access to current, accurate, and consistent data

· It generates adapter metadata as WSDL files with J2CA extension.

9.2. MICROCONTROLLER - ESP32:

INTRODUCTION

Arduino is a great platform for beginners into the World of Microcontrollers and Embedded Systems. With a lot of cheap sensors and modules, you can make several projects either as a hobby or even commercial.

As technology advanced, new project ideas and implementations came into play and one particular concept is the Internet of Things or IoT. It is a connected platform, where several “things” or devices are connected over the internet for exchange of information.

In the DIY community, the IOT projects are mainly focused on Home Automation and Smart Home applications but commercial and industrial IoT projects have far more complex implementations like Machine Learning, Artificial Intelligence, Wireless Sensor Networks etc.

The important thing in this brief intro is whether it is a small DIY project by a hobbyist or a complex industrial project, any IoT project must have connectivity to the Internet. This is where the likes of ESP8266 and ESP32 come into picture.

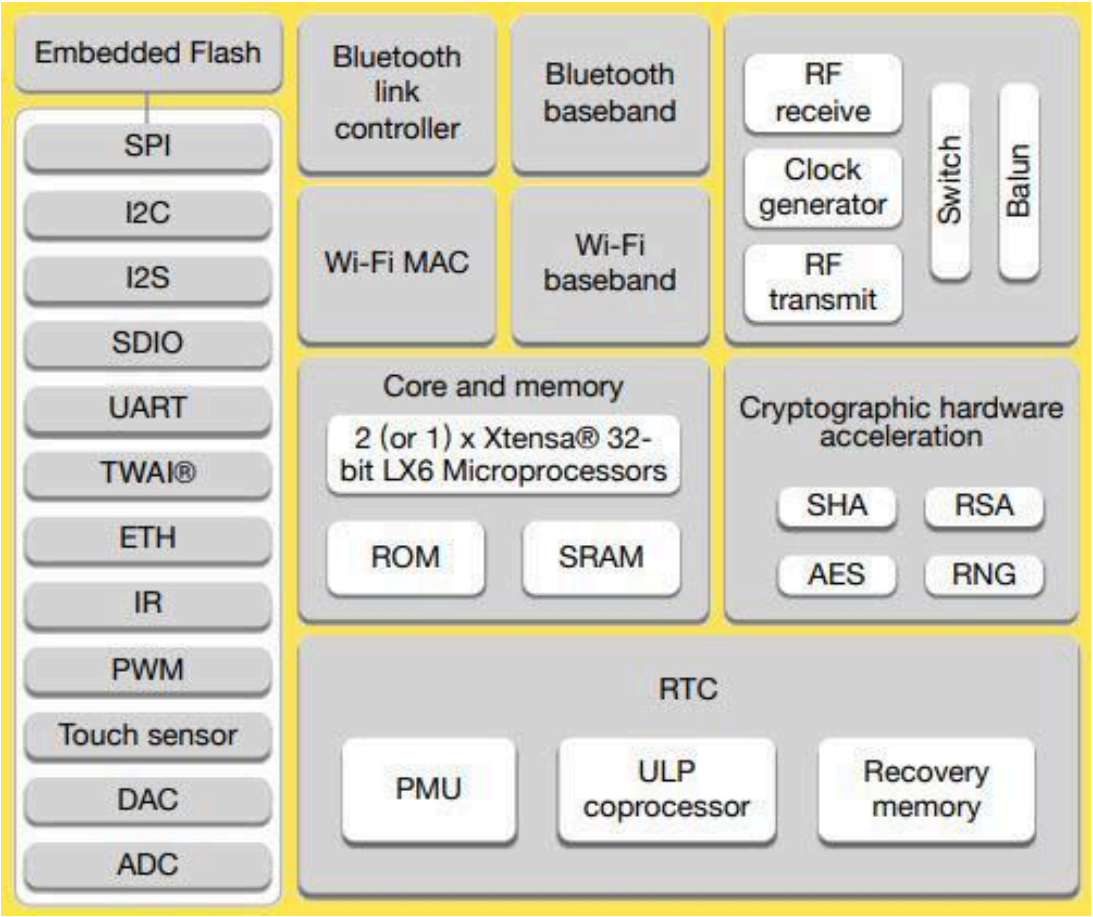
If you want to add Wi-Fi connectivity to your projects, then ESP8266 is a great option. But if you want to build a complete system with Wi-Fi connectivity, Bluetooth connectivity, high resolution ADCs, DAC, Serial Connectivity and many other features, then ESP32 is the ultimate choice.

What is ESP32?

ESP32 is a low-cost System on Chip (SoC) Microcontroller from Espressif Systems, the developers of the ESP8266 SoC. It is a successor to ESP8266 SoC and comes in both single-core and dual-core variations of Tensilica's 32-bit Xtensa LX6 Microprocessor with integrated Wi-Fi and Bluetooth.

The good thing about ESP32, like ESP8266 is its integrated RF components like Power Amplifier, Low-Noise Receive Amplifier, Antenna Switch, Filters and RF Balun. This makes designing hardware around ESP32 very easy as you require very few external components.

Another important thing to know about ESP32 is that it is manufactured using TSMC’s ultra-low-power 40 nm technology. So, designing battery operated applications like wearables, audio equipment, baby monitors, smart watches, etc., using ESP32 should be very easy.



GENERAL DESCRIPTION:

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.

Specifications of ESP32

ESP32 has a lot more features than ESP8266 and it is difficult to include all the specifications in this Getting Started with ESP32 guide. So, I made a list of some of the important specifications of ESP32 here. But for a complete set of specifications, I strongly suggest you refer to the Datasheet.

- Single or Dual-Core 32-bit LX6 Microprocessor with clock frequency up to 240 MHz.
- 520 KB of SRAM, 448 KB of ROM and 16 KB of RTC SRAM.
- Supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps.
- Support for both Classic Bluetooth v4.2 and BLE specifications.
- 34 Programmable GPIOs.

- Up to 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC
- Serial Connectivity include 4 x SPI, 2 x I^2C, 2 x I^2S, 3 x UART.
- Ethernet MAC for physical LAN Communication (requires external PHY).
- 1 Host controller for SD/SDIO/MMC and 1 Slave controller for SDIO/SPI.
- Motor PWM and up to 16-channels of LED PWM.
- Secure Boot and Flash Encryption.
- Cryptographic Hardware Acceleration for AES, Hash (SHA-2), RSA, ECC and RNG.

9.3. IR SENSOR :

IR LED emits infrared radiation. This radiation illuminates the surface in front of the LED. Depending on the reflectivity of the surface, the amount of light reflected varies. This reflected light is made on a reverse biased IR sensor. The amount of electron-hole pairs generated depends on intensity of incident IR radiation. Thus as intensity of incident ray varies, voltage across resistor will vary accordingly.

PRODUCT DESCRIPTION

An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detect the motion. Usually in the infrared spectrum, all the objects radiate some form of thermal radiations. These

types of radiation are invisible to our eyes, and can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, The resistances and these output voltages change in proportion to the magnitude of the IR light received.

FEATURES :

· Operating voltage:5VDC

· Output voltage: 0 or 5VDC

· Easy to assemble and use

· Onboard detection indication

· Effective distance range of 2cm

APPLICATIONS :

· Augmentative communication devices

· Car locking systems

· Computers

· Signage

· Telephones

9.4. RFID READER :

A Radio Frequency Identification Reader (RFID reader) is a device used to gather information from an RFID tag, which is used to track individual objects. Radio Frequency waves are used to transfer data from the tag to a reader. The RFID tag must be within the range of an RFID reader, in order to be read. RFID technology allows several items to be quickly scanned and enables fast identification of a particular product, even when it is surrounded by several other items.

PRODUCT DESCRIPTION :

Radio frequency identification (RFID) is one method for Automatic Identification and Data Capture (AIDC). RFID tags are used in many industries. An RFID system consists of three components: an antenna and transceiver and a transponder. The antenna uses radio frequency waves to transmit a signal that activates the transponder. When activated, the tag transmits data back to the antenna. An RFID reader's function is to interrogate RFID tags. The means of interrogation is wireless and because the distance is relatively short; line of sight between the reader and tags is not necessary. A reader contains an RF module, which acts as both a transmitter and receiver of radio frequency signals. The transmitter consists of an oscillator to create the carrier frequency; a modulator to impinge data commands upon this carrier

signal and an amplifier to boost the signal enough to awaken the tag. The receiver has a demodulator to extract the returned data and also contains an amplifier to strengthen the signal for processing. A microprocessor forms the control unit, which employs an operating system and memory to filter and store the data. The data is now ready to be sent to the network.

RFID Reader

FEATURES:

- Supply voltage: 12v DC
- Output: UART and TTL
- In-built buzzer indicator
- Signal LED is placed

APPLICATIONS

- Passports

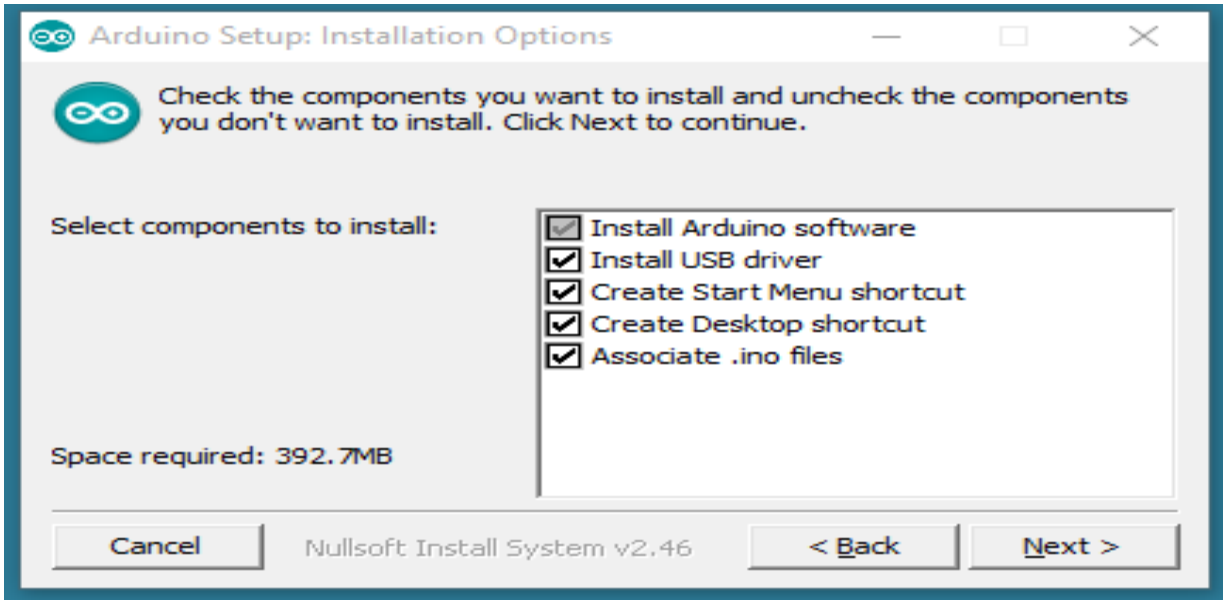
- Toll booth passes
- Hospitals
- Libraries

10. SOFTWARE DESCRIPTION

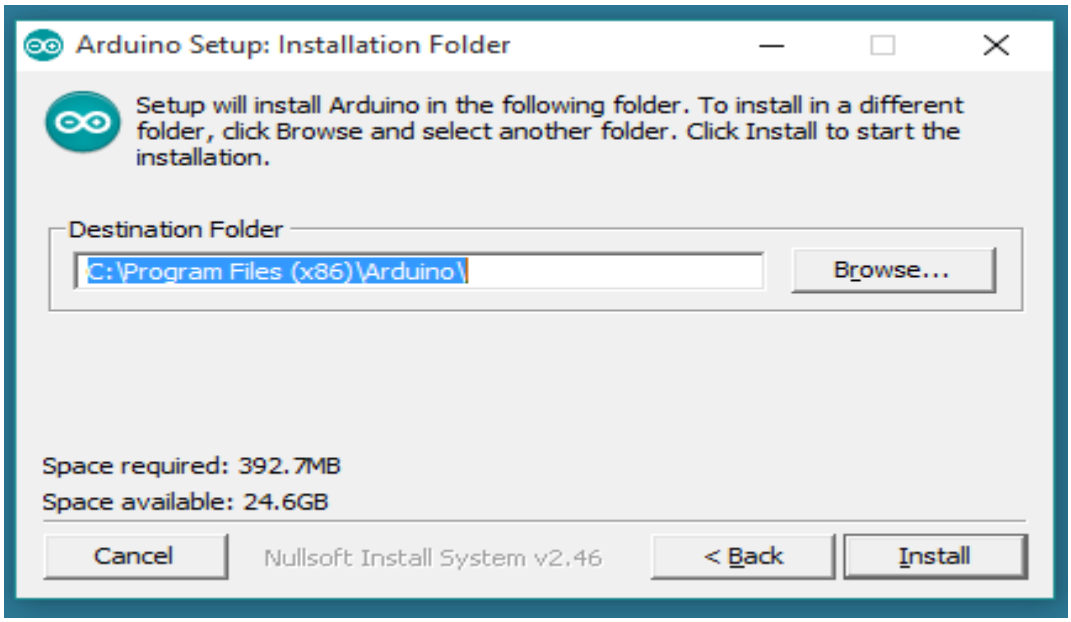
Arduino Software (IDE):

Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.

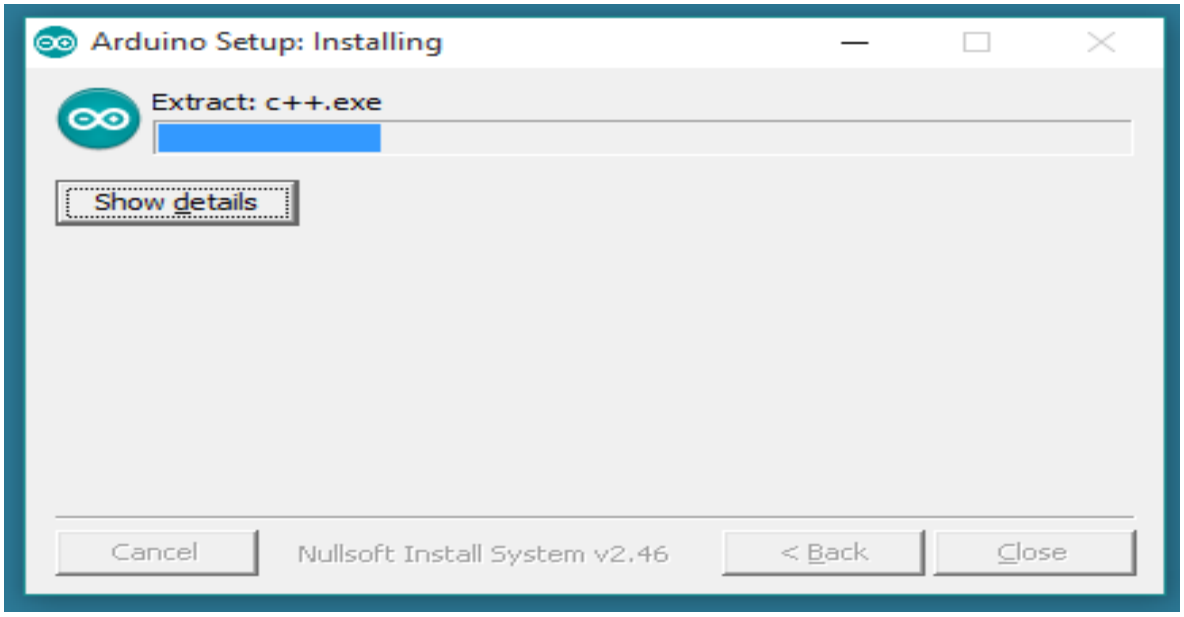
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install



Choose the installation directory (we suggest to keep the default one)



The process will extract and install all the required files to execute properly the Arduino Software (IDE)

Arduino Bootloader Issue

The current boot loader burned onto the Arduino UNO is not compatible with ROBOTC. In its current form, you will be able to download the ROBOTC Firmware to the ArduinoUNO, but you will not able to download any user programs.

The reason for this is because there is a bug in the Arduino UNO firmware that does not allow flash write commands to start at anywhere but the beginning of flash memory (0x000000). See the bottom of this page for more technical details.

Because ROBOTC is not able to burn a new bootloader as of today, you will need to use the Arduino's Open Source language with a modified bootloader file to re-burn your bootloader on your Arduino UNO boards. The enhanced bootloader is backwards compatible with the original one.

That means you'll still be able to program it through the Arduino programming environment as before, in addition to ROBOTC for Arduino.

Hardware Needed:

To burn a new version of the Arduino bootloader to your UNO, you'll need an AVR ISP Compatible downloader.

Using an AVR ISP (In System Programmer)

- Your Arduino UNO (to program)
- An AVR Programmer such as the AVR Pocket Programmer
- An AVR Programming Cable (the pocket programmer comes with one)

If you have extra Arduino boards, but no ISP programmer, SparkFun.com has a cool tutorial on how to flash a bootloader using an Arduino as an ISP.

Using another Arduino as an ISP

- Your Arduino UNO (to program)
- A Working Arduino (doesn't matter what kind)
- Some Male-to-Male Jumper Cables.

Software Needed:

ROBOTC is not currently able to burn a bootloader onto an Arduino board, so you'll need to download a copy of the latest version of the Arduino Open-Source programming language.

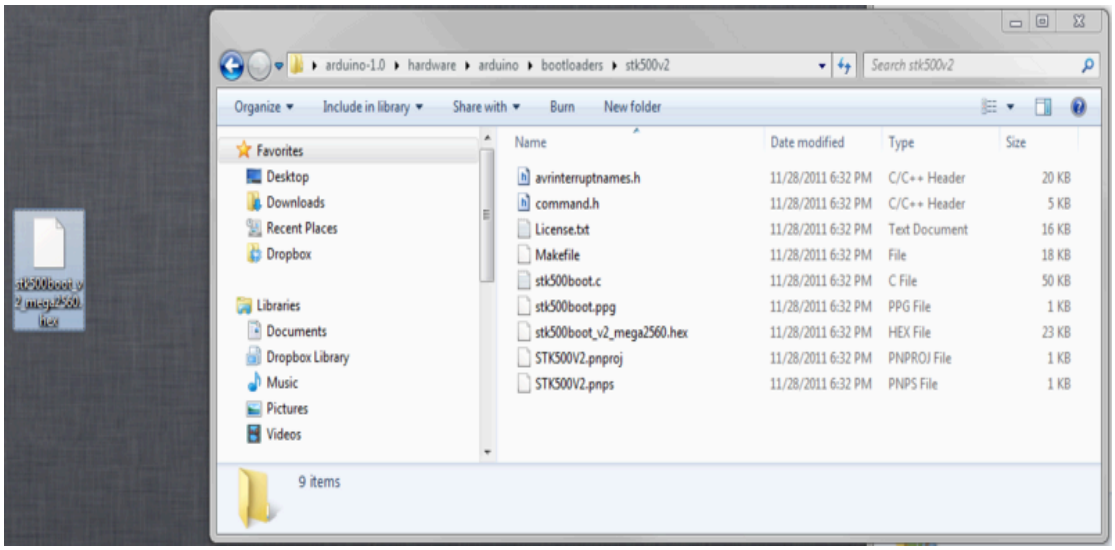
- Arduino Official Programming Language - Download Page

In addition, you'll need the ROBOTC modified bootloader. You can download that here:

- ROBOTC Modified UNO Bootloader - Modified Bootloader

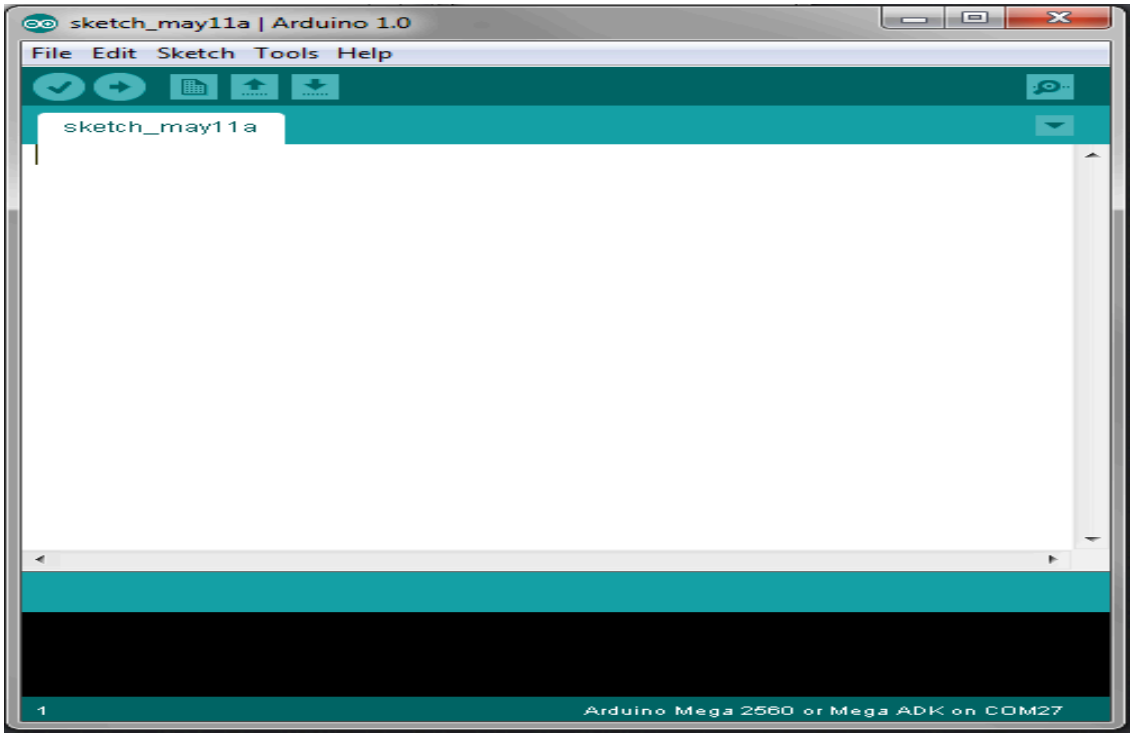
Bootloader Download Instructions

- Download the Arduino Open Source Software and a copy of the Modified Bootloader File
- Copy the Modified Bootloader File into the /Arduino-1.0/hardware/arduino/bootloaders/stk500v2/ and overwrite the existing bootloader.



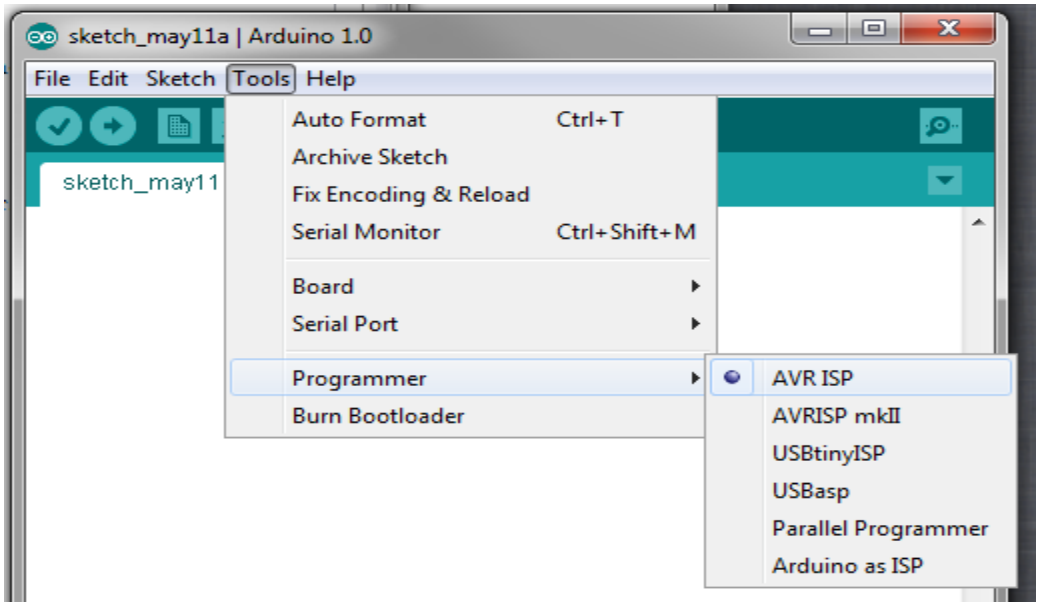
- Power up your Arduino UNO (either via USB or external power)

- Plug in your AVR ISP Programmer to your computer (make sure you have any required drivers installed)
- Connect your AVR ISP Programmer into your Arduino UNO Board via the ISP Header (the 2x3 header pins right above the Arduino Logo)
- Launch the Arduino Open Source Software

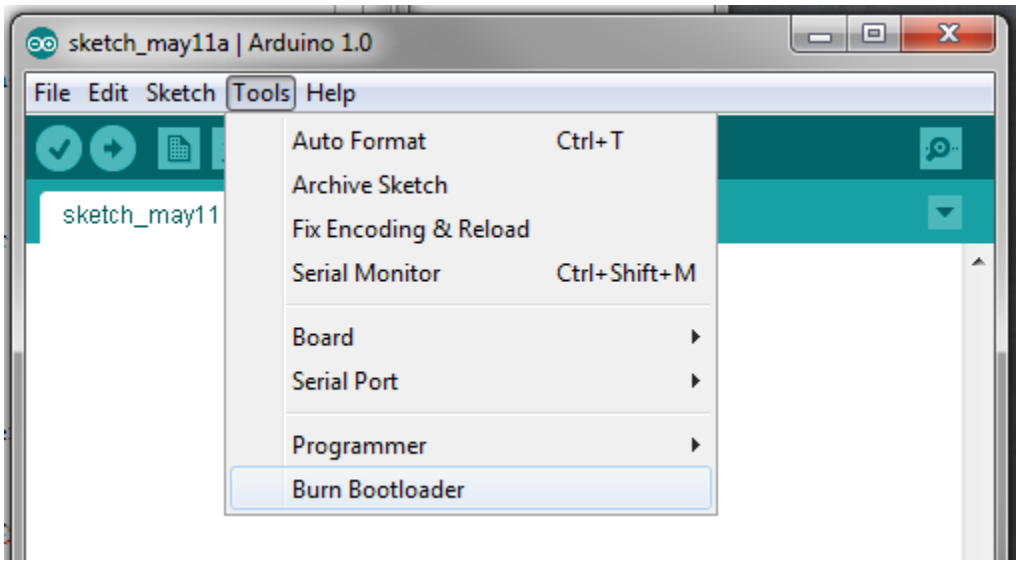


- Change your settings in the Arduino Software to look for an Arduino UNO

- Change your settings in the Arduino Software to select your ISP Programmer Type (Check your programmer's documentation for the exact model)



- Select the "Burn Bootloader" option under the "Tools" menu. The modified bootloader will now be sent to your Arduino. This typically take a minute or so.



- You should be all set to download ROBOTC firmware and start using your Arduino UNO with ROBOTC.

Summary:

Microcontroller	Arduino UNO
Operating Voltage	5V Input Voltage (recommended)
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40mA
DC Current for3.3VPin	50mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8KB
EEPROM	4KB
Lock Speed	16MHz

The Arduino UNO can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and V_{in} pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the

voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

They differ from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the program as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via a non-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.**A3.3voltsupplygeneratedbytheon-boardregulator.Maximumcurrentdrawis50mA.
- **GND.** Ground pins.

The ATMEGA has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50k Ohms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATMEGA USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a changing value. See the attach Interrupt() function for details.
- **PWM: 0to13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Arduino UNO has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analog Reference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analog Reference().
- **eset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

11. CONCLUSION:

The intelligent traffic light management system proposed in this project offers a practical and effective solution to urban traffic problems. By using IR sensors for traffic density detection and RFID for emergency vehicle prioritization, the system ensures smoother traffic flow and faster emergency response times. It eliminates the drawbacks of traditional fixed-time traffic signals and manual interventions. This project presents a scalable, low-cost, and efficient solution that can play a significant role in the development of smart transportation infrastructure in modern cities.