INTELLIGENT FIRE FIGHTING ROBOT WITH DEEP LEARNING

A PROJECT REPORT

Submitted by

LOKESH.V	**(211419105073)**
MOHAMMED ARIF.J.S	**(211419105080)**
DHILIP KUMAR.P	**(211419105030)**
HARISH DEV.R	**(211419105044)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **“INTELLIGENT FIRE - FIGHTING ROBOT WITH DEEP LEARNING** “is the bonafide work of **“LOKESH.V(211419105073), MOHAMMED ARIF.J S (211419105080), DHILIP KUMAR.P (211419105030), HARISH DEV.R (211419105044)”** who carried out the project under my supervision.

Signature of the HOD	Signature of the Supervisor
Dr. S. Selvi	Mr. S. Balaji
PROFESSOR	SUPERVISOR
HEAD OF THE DEPARTMENT	ASSISTANT PROFESSOR,
Department of Electrical and	Department of Electrical and
Electronics Engineering,	Electronic Engineering,
Panimalar Engineering College,	Panimalar Engineering
College,	
Chennai – 123	Chennai – 123

Submitted for End Semester Project Viva Voce held on......................... at Panimalar Engineering College, Chennai.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Our sincere thanks to our Honourable Founder and Chairman, **Dr.JEPPIAAR,M.A.,B.L.,Ph.D.,** for his sincere endeavour in educating us in his premier institution.

We would like to express our deep gratitude to our beloved **Secretary and Correspondent, Dr.P.CHINNADURAI, M.A.,M.Phil.,Ph.D** for his enthusiastic motivation which inspired us a lot in completing this project and our sincere thanks to our Directors **Mrs.C.VIJAYA RAJESWARI, Dr.C.SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYASREE SAKTHIKUMAR,B.E,M.B.A,Ph.D** for providing us with the necessary facilities for the completion of this project.

We would like to express thanks to our Principal, **Dr. K. Mani M.E., Ph.D.,** for having extended his guidance and cooperation.

We would also like to thank our **Head of the Department**, **Dr.S. Selvi, M.E., Ph.D., Professor and Head, Department of Electrical and Electronics Enginering** for her encouragement.

Personally, we thank our Guide **Mr. S. Balaji, M.E., Professor, Department of Electrical and Electronics engineering** for the persistent motivation and support for this project, who at all times was the mentor of germination of the project from a small idea.

We express our sincere thanks to the project coordinators **Dr.S.Deepa & Dr.N.MANOJ KUMAR,M.E.,Ph.D., in Department of Electrical and Electronics Engineering** for the Valuable suggestions from time to time at every stage of our project.

Finally, we would like to take this opportunity to thank our family members, faculty and non-teaching staff members of our department, friends, well-wishers who have helped us for the successful completion of our project.

ABSTRACT

It is known that the technological advancements are increasing at a faster pace. But the utilization of technologies in various sectors are very low. So we propose an automated robotic system where it automatically senses an occurrence of fire by using flame sensor. The robot takes controlling section by itself based on the user predefined command. By using flame sensor it will continuously monitoring If any fire accident occur or not. Even if any fire accident occurs it will automatically activate a water spraying mechanism. The system provides an automated fire monitoring and clearance system. Flames are typically seen when they have spread over a huge region, making its control and stoppage difficult. The result is loss of men and material resources and it causes harm to the earth and air as well. The fire may start due to human activities like smoking or barbecue parties or due to normal reasons, like, very high temperature conditions during summer season or there may be a wrecked glass that acts as a focal point concentrating the daylight on a little spot for a time span, thus prompting fire-start. When fire begins, combustible materials available in the surrounding area act as a fuel leading to bigger and wider fires. Timely detection of occurrence of fire can significantly lessen the impending harm as well as the expense on fire fighting.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

ADC – Analog To Digital Converter

PWM – Pulse Width Modulation

SCL – Serial Clock

SPI - SerialPeripheral Interface

I2C – Inter Integrated Circuit

DAC – Digital To Analog Converter

MISO – Multiple Input Single Output

MOSI - Multiple Output Single Input

UAV - Unmanned Aerial Vehicle

CHAPTER 1

INTRODUCTION

1.1. OBJECTIVE

To design and implement a fire fighting robot.

1.2 INTRODUCTION

Flames are typically seen when they have spread over a huge region, making its control and stoppage difficult. The result is loss of men and material resources and it causes harm to the earth and air as well. The fire may start due to human activities like smoking or barbecue parties or due to normal reasons, like, very high temperature conditions during summer season or there may be a wrecked glass that acts as a focal point concentrating the daylight on a little spot for a time span, thus prompting fire-start. When fire begins, combustible materials available in the surrounding area act as a fuel leading to bigger and wider fires. Timely detection of occurrence of fire can significantly lessen the impending harm as well as the expense on fire fighting. Forest fires usually appear in complicated terrain, as a result, ground vehicles are obviously inaccessible to these fire areas. In recent years, Unmanned Aerial Vehicles (UAVs) have been paid increasing attention and become a very promising solution to forest fires searching and monitoring. This paper explores the application of UAVs in the wild forest fires searching and fire frontier monitoring mission. First, the FARSITE fire model is used to simulate the realistic wild fire behaviour. After that, the fire search problem is solved with Genetic Algorithm (GA). Then, a cooperative fire monitoring algorithm is investigated. Finally, simulation results show the effectiveness of the proposed searching and monitoring methods.

CHAPTER 2

LITERATURE SURVEY

2.1. COOPERATIVE FOREST MONITORING AND FIRE DETECTION USING A TEAM OF UAVS-UGVS

ABSTRACT: This paper investigates forest monitoring and fire detection strategies using a team of unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs). UAVs have great advantages in forest fire detection and fighting. However they have limited running time and payload capabilities, therefore UGVs are paired with UAVs to avoid their demerits. First, UGVs are used to transport the UAVs to the nearest location to their assigned search area. The UAVs will take-off and start the monitoring and search mission. When one of the UAVs detects the fire, it sends the fire spot coordinates to the leader UGV and ground fire management personnel. Then, the leader UGV that has a powerful computing and image processing capabilities will generate the reference trajectory for UAVs to follow in order to detect and continuously monitor the fire spread. Simulation results are presented in order to demonstrate the effectiveness of the proposed algorithm.

AUTHOR: Khaled_A. Ghamry; Mohamed A. Kamel; Youmin Zhang

PUBLISHED IN: 2016 International Conference on Unmanned Aircraft Systems (ICUAS)

2.2. A STUDY ON PATH PLANNING FOR SMALL MOBILE ROBOT TO MOVE IN FOREST AREA

ABSTRACT: We are developing an autonomous monitoring system using mobile robot in response to the demands of autonomous monitoring in forest area.

The effective path planning is required for autonomous operation. The robot needs to locomote in the grassy area so as to move in a natural forest area, therefore the path route on grassy area have to be considered. The objective of this study was to develop a path planning method for small mobile robot to move in the forest. We focused on the cost used to generate the path and try to add grass vegetation into the cost map. The grass vegetation degree is effective to generate the optimal path, and the robot could move in forest by following the generated path.

AUTHOR: K. Tanaka; Y. Okamoto; H. Ishii; D. Kuroiwa; H. Yokoyama; S. Inoue; Q. Shi; S. Okabayashi; Y. Sugahara; A. Takanishi

PUBLISHED IN: 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)

2.3. FIRE-FRONT RECOGNITION IN UAV-BASED FOREST-FIRE MONITORING SYSTEM USING FUZZY ROUGH SOFT SETS

ABSTRACT: This work describes a new method to recognize the forest-fire front based on uncertain observations in a forest fire-fighting monitoring system, which jointly uses UAVs and remote sensing. A method of the fire front representation based on soft fuzzy rough level sets is proposed, the required image processing and remote sensing algorithms are presented. The proposed method has acceptable computational complexity in a real-time forest fire response decision support system**.**

AUTHOR: Vladimir Sherstjuk; Maryna Zharikova

PUBLISHED IN: 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)

2.4. A SOLUTION FOR SEARCHING AND MONITORING FOREST FIRES BASED ON MULTIPLE UAVS

ABSTRACT: Forest fires usually appear in complicated terrain, as a result, ground vehicles are obviously inaccessible to these fire areas. In recent years, Unmanned Aerial Vehicles (UAVs) have been paid increasing attention and become a very promising solution to forest fires searching and monitoring. This paper explores the application of UAVs in the wild forest fires searching and fire frontier monitoring mission. First, the FARSITE fire model is used to simulate the realistic wild fire behavior. After that, the fire search problem is solved with Genetic Algorithm (GA). Then, a cooperative fire monitoring algorithm is investigated. Finally, simulation results show the effectiveness of the proposed searching and monitoring methods.

AUTHOR: Yintao Zhang; Youmin Zhang; Ziquan Yu

PUBLISHED IN: 2019 International Conference on Unmanned Aircraft Systems (ICUAS)

2.5. UNMANNED AERIAL VEHICLE BASED FOREST FIRE MONITORING AND DETECTION USING IMAGE PROCESSING TECHNIQUE

ABSTRACT: Early forest fire alarm systems are critical in making prompt response in the event of unexpected hazards. Cost-effective cameras, improvements in memory, and enhanced computation power have all enabled the design and real-time application of fire detecting algorithms using light and small-size embedded surveillance systems. This is vital in situations where the performance of traditional forest fire monitoring and detection techniques are unsatisfactory. This paper presents a forest fire monitoring and detection method with visual sensors onboard unmanned aerial vehicle (UAV). Both color and motion features of fire are adopted for the design of the studied forest fire detection strategies. This is for the purpose of improving fire detection performance, while reducing false alarm rates. Indoor experiments are conducted to demonstrate the effectiveness of the studied forest fire detection methodologies.

AUTHOR: Chi Yuan; Khaled A. Ghamry; Zhixiang Liu; Youmin Zhang

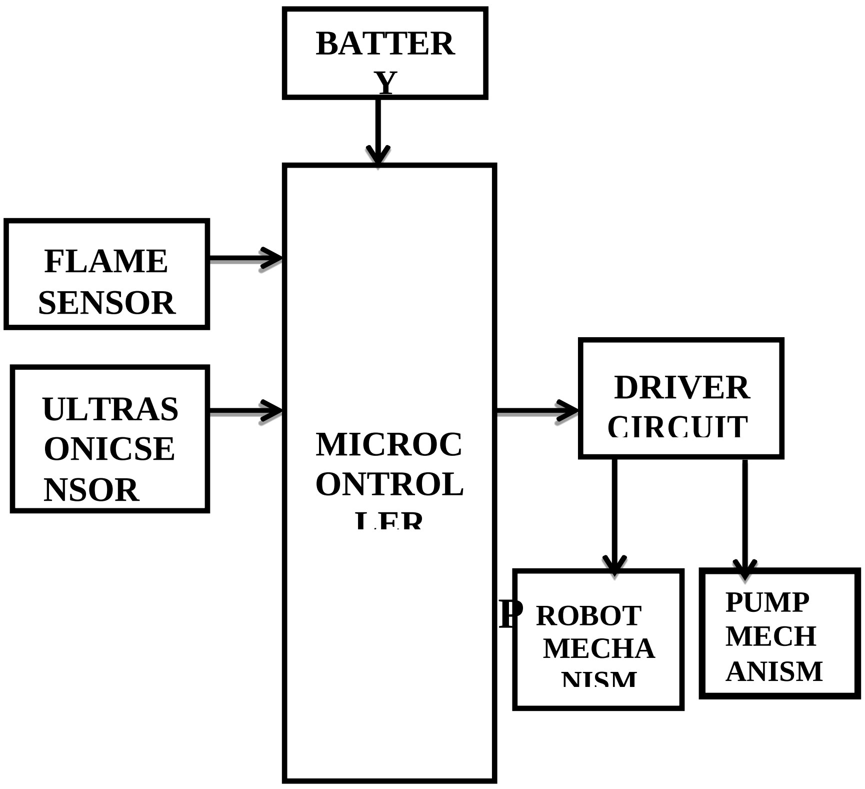
PUBLISHED IN: 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)

CHAPTER 3
OVERVIEW

3.1 PROPOSED SYSTEM

The proposed system here overcomes the disadvantages explained by the existing system. We are proposing a robot for fire monitoring in the dense forest. We are implementing a robot for certain distance which will keep monitoring by using flame sensor. If any fire occurrence once the robot detects the fire it will activate the water pump to spray the water available in the tank which will be fixed in the robot.

3.2 BLOCK DIAGRAM



HARDWARE COMPONENTS

4.1. HARDWARE REQUIREMENTS

- BATTERY
- MICROCONTROLLER
- FLAME SENSOR
- ULTRASONIC SENSOR
- DRIVER CIRCUIT
- DC MOTOR
- PUMP MOTOR

4.2. HARDWARE DESCRIPTIONS

4.2.1. Battery

In electricity, a battery is a device consisting of one or more electrochemical cells that convert stored chemical energy into electrical energy. Since the invention of the first battery (or "voltaic pile") in 1800 by Alessandro Volta and especially since the technically improved Daniell cell in 1836, batteries have become a common power source for many household and industrial applications. According to a 2005 estimate, the worldwide battery industry generates US$48 billion in sales each year, with 6% annual growth.

There are two types of batteries: primary batteries (disposable batteries), which are designed to be used once and discarded, and secondary batteries (rechargeable batteries), which are designed to be recharged and used multiple times. Batteries come in many sizes from miniature cells used to power hearing aids and wristwatches to battery banks the size of rooms that provide standby power for telephone exchanges and computer data centers.

A battery is a device that converts chemical energy directly to electrical energy. It consists of a number of voltaic cells; each voltaic cell consists of two half-cells connected in series by a conductive electrolyte containing anions and cations. One half-cell includes electrolyte and the electrode to which anions (negatively charged ions) migrate, i.e., the anode or negative electrode; the other half-cell includes electrolyte and the electrode to which cations (positively charged ions) migrate, i.e., the cathode or positive electrode. In the redox reaction that powers the battery, cations are reduced (electrons are added) at the cathode, while anions are oxidized (electrons are removed) at the anode. The electrodes do not touch each other but are electrically connected by the electrolyte. Some cells use two half-cells with different electrolytes. A separator between half-cells allows ions to flow, but prevents mixing of the electrolytes.

Batteries are classified into two broad categories, each type with advantages and disadvantages.

- **Primary batteries** irreversibly (within limits of practicality) transform chemical energy to electrical energy. When the initial supply of reactants is exhausted, energy cannot be readily restored to the battery by electrical means.
- **Secondary batteries** can be recharged; that is, they can have their chemical reactions reversed by supplying electrical energy to the cell, restoring their original composition.

Some types of primary batteries used, for example, for telegraph circuits, were restored to operation by replacing the components of the battery consumed by the chemical reaction. Secondary batteries are not indefinitely rechargeable due to dissipation of the active materials, loss of electrolyte and internal corrosion.

Primary batteries

Main article: Primary cell

Primary batteries can produce current immediately on assembly. Disposable batteries are intended to be used once and discarded. These are most commonly used in portable devices that have low current drain, are used only intermittently, or are used well away from an alternative power source, such as in alarm and communication circuits where other electric power is only intermittently available. Disposable primary cells cannot be reliably recharged, since the chemical reactions are not easily reversible and active materials may not return to their original forms. Battery manufacturers recommend against attempting recharging primary cells. Common types of disposable batteries include zinc–carbon batteries and alkaline batteries. In general, these have higher energy densities than rechargeable batteries, but disposable batteries do not fare well under high-drain applications with loads under 75 ohms (75 Ω).

Secondary batteries

Main article: Rechargeable battery

Secondary batteries must be charged before use; they are usually assembled with active materials in the discharged state. Rechargeable batteries or **secondary cells** can be recharged by applying electric current, which reverses the chemical reactions that occur during its use. Devices to supply the appropriate current are called chargers or rechargers.

The oldest form of rechargeable battery is the lead–acid battery. This battery is notable in that it contains a liquid in an unsealed container, requiring that the battery be kept upright and the area be well ventilated to ensure safe dispersal of the hydrogen gas produced by these batteries during overcharging. The lead–acid battery is also very heavy for the amount of electrical energy it can supply. Despite this, its low manufacturing cost and its high surge current levels make its use common where a large capacity (over approximately 10 Ah) is required or where the weight and ease of handling are not concerns.

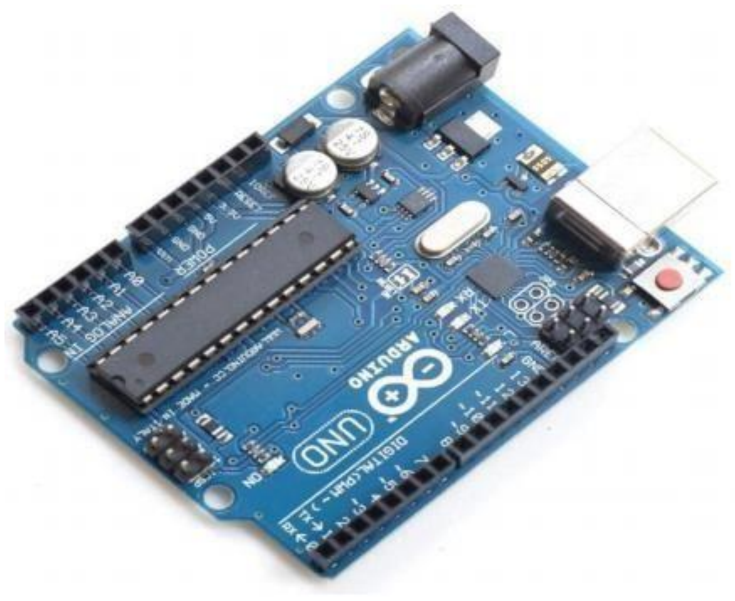
4.2.2. Microcontroller Arduino Uno

4.2.2.1. General Description

Arduino is an open-source project that created microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices. The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. These systems provide sets of digital and analog input/output (I/O) pins that can interface to various expansion boards (termed shields) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) based on a programming language named Processing, which also supports the languages C and C++.

4.2.2.2. Product Description

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter. Arduino Uno has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers.



Arduino UNO

Fig 1.1

4.2.2.3. Features

- Microcontroller: ATmega328P
- Operating voltage: 5V
- Input voltage: 7-12V
- Flash memory: 32KB
- SRAM: 2KB
- EEPROM: 1KB

4.2.2.4. Applications

- Real time biometrics
- Robotic applications
- Academic applications

4.2.3. Flame Sensor

A sensor which is most sensitive to a normal light is known as a flame sensor. That’s why this sensor module is used in flame alarms. This sensor detects flame otherwise wavelength within the range of 760 nm – 1100 nm from the light source. This sensor can be easily damaged to high temperature. So this sensor can be placed at a certain distance from the flame. The flame detection can be done from a 100cm distance and the detection angle will be 600. The output of this sensor is an analog signal or digital signal. These sensors are used in fire fighting robots like as a flame alarm. A flame-sensor is one kind of detector which is mainly designed for detecting as well as responding to the occurrence of a fire or flame. The flame detection response can depend on its fitting.

4.2.3.1. Working Principle

This sensor/detector can be built with an electronic circuit using a receiver like electromagnetic radiation. This sensor uses the infrared flame flash method, which allows the sensor to work through a coating of oil, dust, water vapour, otherwise ice.

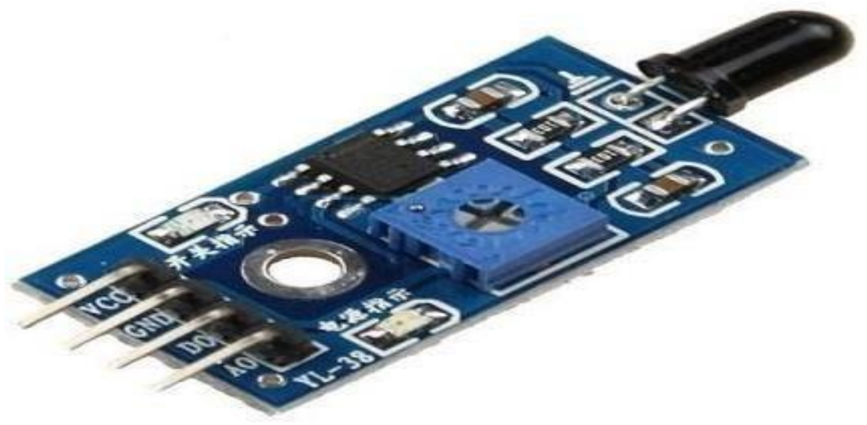


FIG 1.2

4.2.3.2. Features

- Photosensitivity is high
- Response time is fast
- Simple to use

- Sensitivity is adjustable
- Detection angle is 600,
- It is responsive to the flame range.
- Accuracy can be adjustable
- Operating voltage of this sensor is 3.3V to 5V
- Analog voltage o/ps and digital switch o/ps
- The PCB size is 3cm X 1.6cm
- Power indicator & digital switch o/p indicator
- If the flame intensity is lighter within 0.8m then the flame test can be activated, if the flame intensity is high, then the detection of distance will be improved.

4.2.3.3. Applications

- Hydrogen stations
- Industrial heating
- Fire detection
- Fire alarm
- Fire fighting robot
- Drying systems
- Industrial gas turbines
- Domestic heating systems
- Gas-powered cooking devices

4.2.4. Ultrasonic Sensor

4.2.4.1. General Description

Ultrasonic sensor emit ultrasonic pulses, and by measuring the time of ultrasonic pulse reaches the object and back to the transducer. The sonic waves emitted by the transducer are reflected by an object and received back in the transducer. After having emitted the sound waves, the ultrasonic sensor will switch to receive mode. The time elapsed between emitting and receiving is proportional to the distance of the object from the sensor.

4.2.4.2. Product Description

Ultrasonic transmitter emitted an ultrasonic wave in one direction and started timing when it launched. Ultrasonic spread in the air and would return immediately when it encountered obstacles on the way. At last the ultrasonic receiver would stop timing when it receives the reflected wave. The distance of sensor from the target object is calculated. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It operation is not affected by sunlight or black material. The supply voltage to the sensor is 5VDC. The sensor has two pins namely trig and echo which is connected to the controller to give digital input.

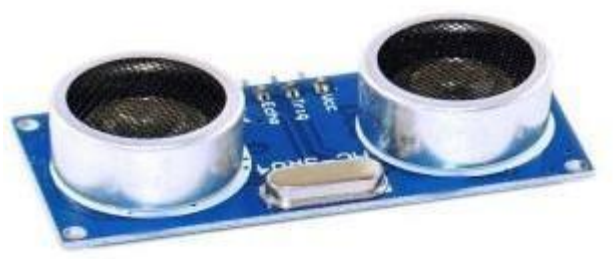


FIG 1.3

4.2.4.3. Features

- Working Voltage: 5VDC
- Quiescent Current: <2mA
- Working Current: 15mA
- Detecting Range: 2cm - 4.5m
- Trigger Input Pulse width: 10uS

4.2.4.4. Applications

- Robot navigation

- Obstacle avoidance
- Engineering measurement tools
- Industrial control system

4.2.5. Driver Circuit

The ULN2003 is a monolithic high voltage and high current Darlington transistor arrays. It consists of seven NPN Darlington pairs that feature high-voltage outputs with common-cathode clamp diode for switching inductive loads. The collector-current rating of a single Darlington pair is 500mA. The darlington pairs may be paralleled for higher current capability. Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED gas discharge), line drivers, and logic buffers.

The ULN2003 has a 2.7kW series base resistor for each Darlington pair for operation directly with TTL or 5V CMOS devices.

4.2.5.1. Features

- 500mA rated collector current (Single output)
- High-voltage outputs: 50V
- Inputs compatible with various types of logic.
- Relay driver application

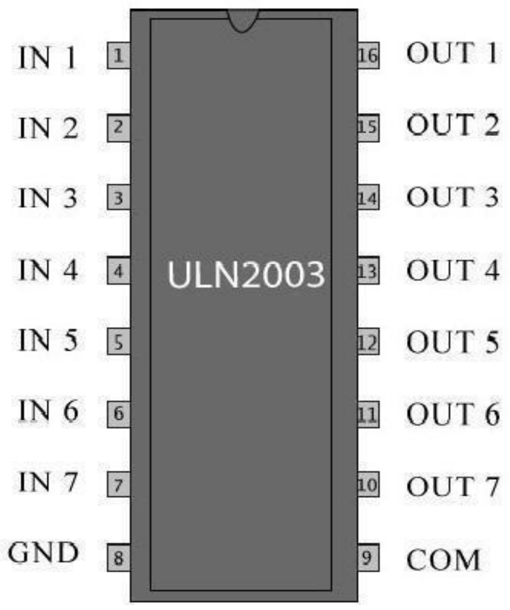
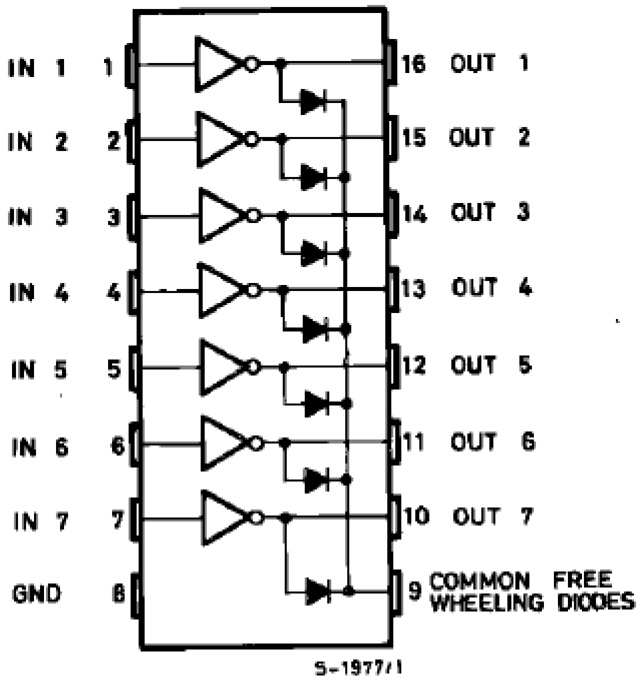


FIG 1.4

Pin Description:

Pin No	Function	Name
1	Input for 1st channel	Input 1
2	Input for 2nd channel	Input 2
3	Input for 3rd channel	Input 3
4	Input for 4th channel	Input 4
5	Input for 5th channel	Input 5
6	Input for 6th channel	Input 6
7	Input for 7th channel	Input 7
8	Ground (0V)	Ground
9	Common free wheeling diodes	Common
10	Output for 7th channel	Output 7
11	Output for 6th channel	Output 6
12	Output for 5th channel	Output 5
13	Output for 4th channel	Output 4
14	Output for 3rd channel	Output 3
15	Output for 2nd channel	Output 2
16	Output for 1st channel	Output 1



The ULN2003 series input resistors selected for operation directly with 5 V TTL or CMOS. These devices will handle numerous interface needs particularly those beyond the capabilities of standard logic buffers. The ULN2003 have series input resistors for operation directly from 6 V to 15 VCMOS or PMOS logic outputs. The ULN 2003 is the standard Darlington arrays.

The outputs are capable of sinking 500mA and will withstand at least 50 V in the OFF state. Outputs may be paralleled for higher load current capability. The ULx2823A/LW and ULx2824A/LW will withstand 95 V in the OFF state.These Darlington arrays are furnished in 18-pin dual in-line plastic packages (suffix ‘A’) or 18-lead small-outline plastic packages (suffix ‘LW’). All devices are pinned with outputs opposite inputs to facilitate ease of circuit board layout. Prefix ‘ULN’ devices are rated for operation over the temperature range of -20□C to +85□C; prefix ‘ULQ’ devices are rated for operation to -40 C□

4.2.6. DC Motor

4.2.6.1. General Description

The relationship between torque vs speed and current is linear as shown left; as the load on a motor increases, Speed will decrease. The graph pictured here represents the characteristics of a typical motor. As long as the motor is used in the area of high efficiency (as represented by the shaded area) long life and good performance can be expected. However, using the motor outside this range will result in high temperature rises and deterioration of motor parts. A motor's basic rating point is slightly lower than its maximum efficiency point. Load torque can be determined by measuring the current drawn when the motor is attached to a machine whose actual load value is known.

4.2.6.2. Product Description

Geared dc motors can be defined as an extension of dc motors. A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM .The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. A DC motor can be used at a voltage lower than the rated voltage. But, below 1000 rpm, the speed becomes unstable, and the motor will not run smoothly.

The relationship between torque vs speed and current is linear as shown left; as the load on a motor increases, Speed will decrease. The graph pictured here represents the characteristics of a typical motor. As long as the motor is used in the area of high efficiency (as represented by the shaded area) long life and good performance can be expected. However, using the motor outside this range will result in high temperature rises and deterioration of motor parts. A motor's basic rating point is slightly lower than its maximum efficiency point. Load torque can be determined by measuring the current drawn when the motor is attached to a machine whose actual load value is known.

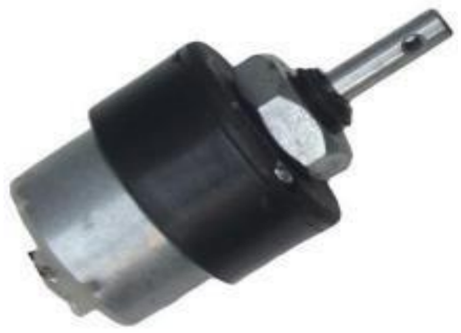


FIG 1.5 DC MOTOR

4.2.6.3. Features

- Supply voltage: 12VDC
- Speed: 60rpm
- Long Lifetime, Low Noise, Smooth Motion
- Equipped with high efficiency

4.2.6.4. Applications

- Coin Changing equipment
- Peristaltic Pumps
- Damper Actuators
- Fan Oscillators
- Photo copier
- Ticket printer

4.2.7. Pump Motor

4.2.7.1. General Description

As the name implies, water pumps pump water. Whether that be in a vehicle, at a business, in the home, or in a well, shoppers can probably find a water pump to fit their vehicle or to help them draw water from the ground in a self-dug well to be used in pressure tanks within the location. Vehicle water pumps help regulate the flow of water through a vehicle's cooling system; when the seal on these go bad, the whole pump must be replaced. Located within the home or business, pressure water pumps regulate the water pressure year round, controlling water flow to different areas of the location.

4.2.7.2. Product Description

A pump motor is a DC motor device that moves fluids. A DC motor converts direct current electrical power into mechanical power. DC or direct current motor works on the principal, when a current carrying conductor is placed in a magnetic field, it experiences a torque and has a tendency to move. This is known as motoring action. Pumps operate by some mechanism (typically reciprocating or rotary), and consume energy to perform mechanical work by moving the fluid. Pumps operate via many energy sources, including manual operation, electricity, engines, or wind power, come in many sizes, from microscopic for use in medical applications to large industrial pumps.

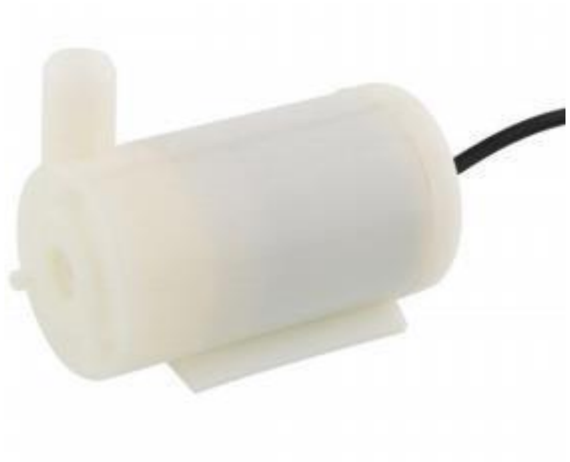


FIG 1.6 PUMP MOTOR

4.2.7.3. Features

- Reduced noise
- Available in DC and AC

- Supply voltage: +12VDC
- Supply voltage: 230V AC

4.2.7.4. Applications

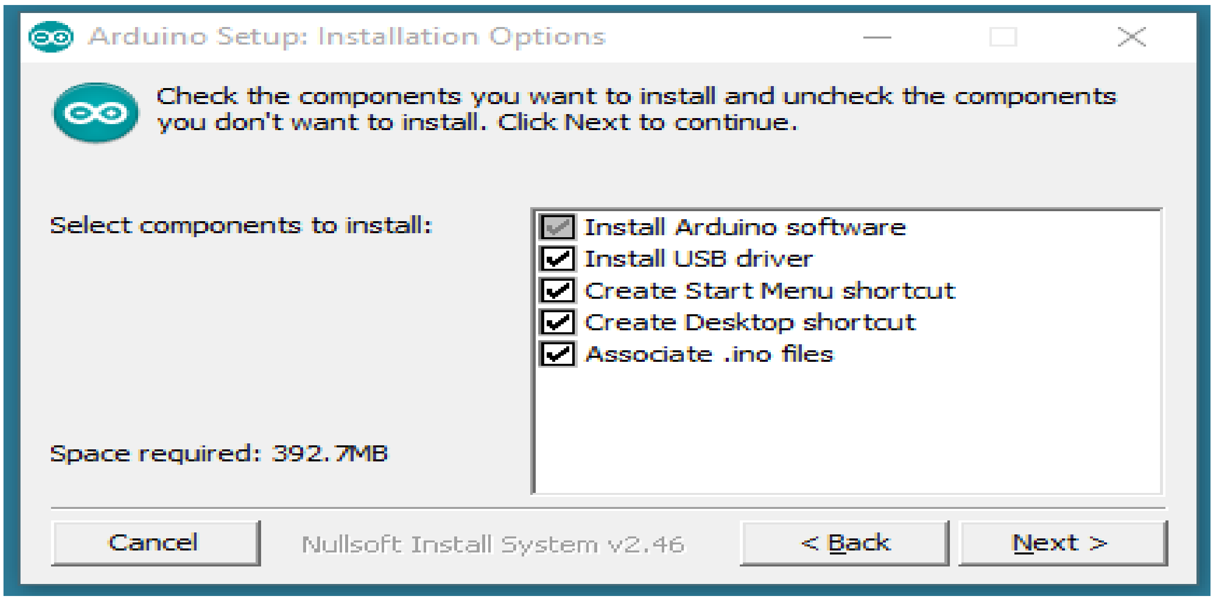
- Priming a pump
- To pump water supply, including pneumatic systems and in places where no suction lift is required
- Pump motor as public water supply
- In domestic water supply system

CHAPTER 5

SOFTWARE COMPONENTS

5.1. 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.



Arduino Setup: Installation Options

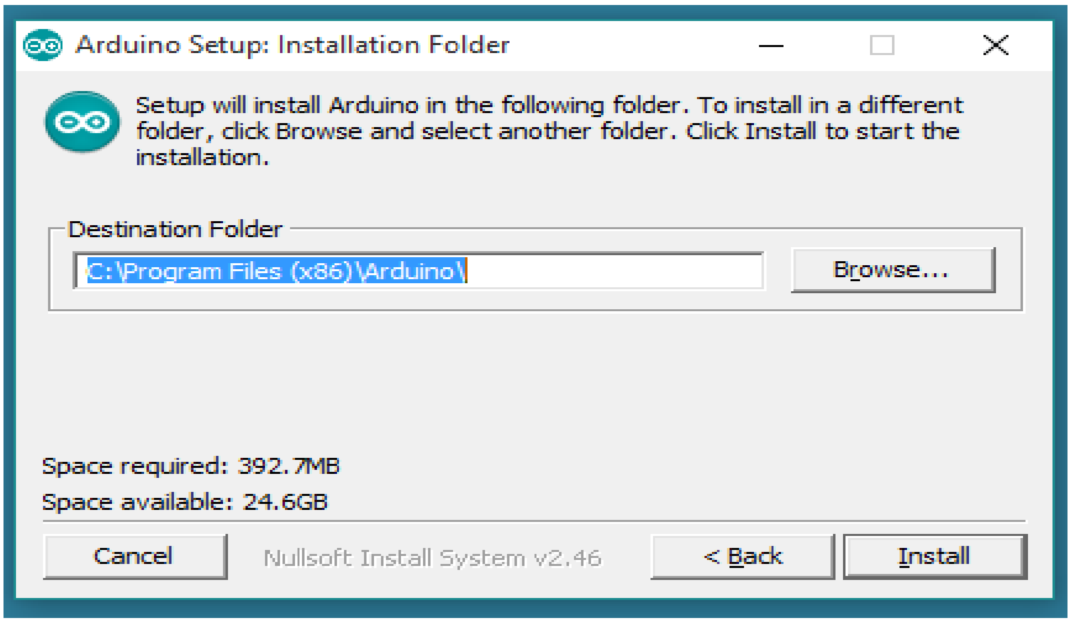
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 392.7MB

FIG 1.7



Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

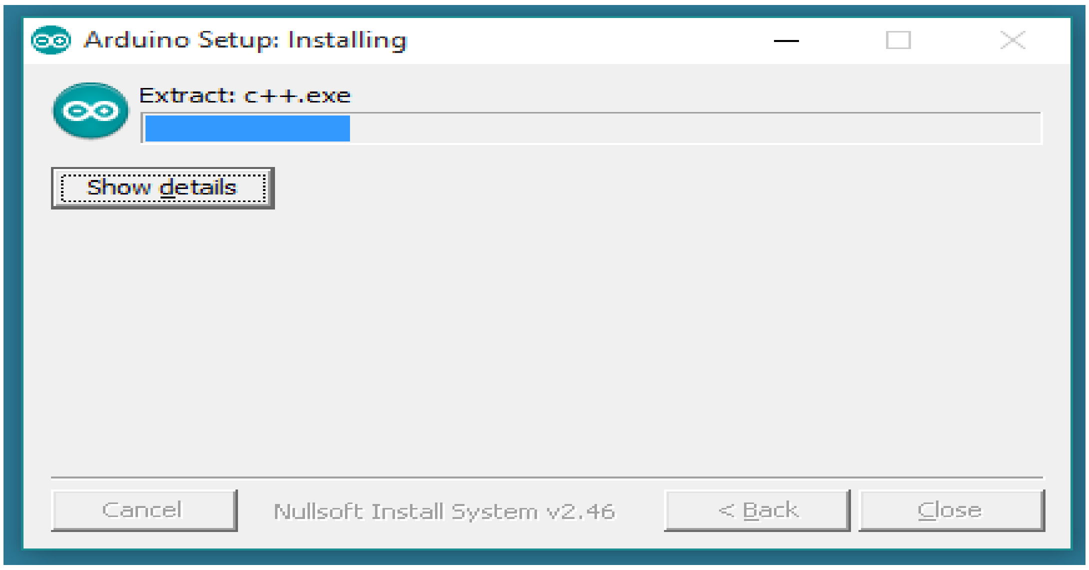
Destination Folder

C:\Program Files (x86)\Arduino\

Space required: 392.7MB

Space available: 24.6GB

FIG 1.8



Extract: c++.exe

Nullsoft Install System v2.46

FIG 1.9

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

5.1.1. Arduino Boot Loader 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.

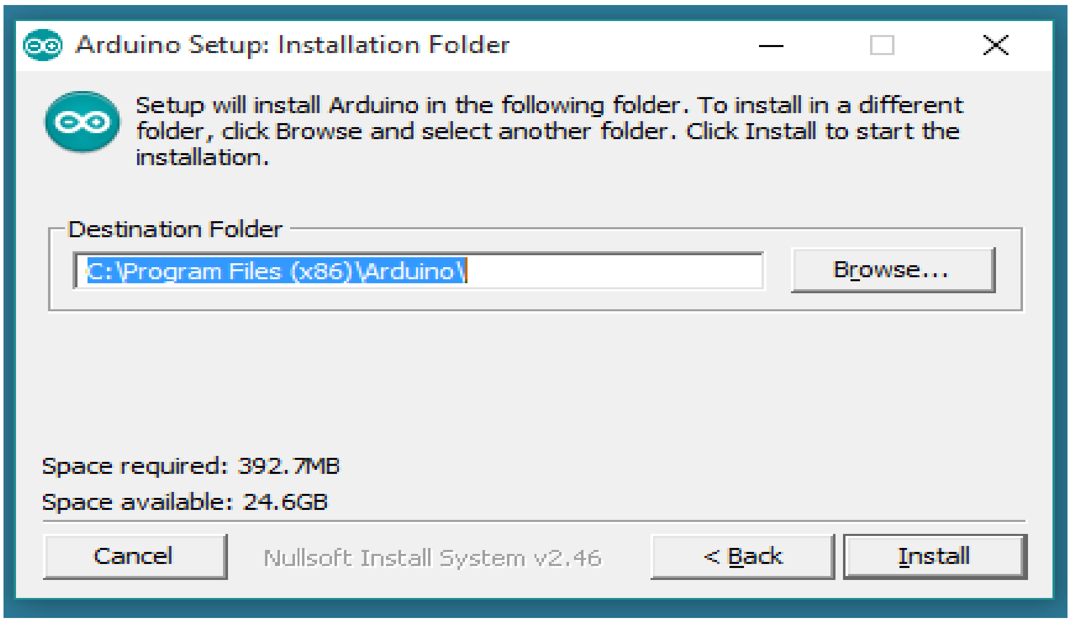
Because ROBOTC is not able to burn a new bootloader as of today, you will need to use the Arduino's Open Source language with a modified bootloader file to re-burn your bootloader on your Arduino UNO boards. The enhanced bootloader is backwards compatible with the original one. That means you'll still be able to program it through the Arduino programming environment as before, in addition to ROBOTC for Arduino.

5.1.2. Hardware Needed

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

5.1.3. Using an AVR ISP (In System Programmer)

- Arduino UNO (to program)



Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

Space required: 392.7MB
Space available: 24.6GB

Nullsoft Install System v2.46

- ISP Header (the 2x3 header pins right above the Arduino Logo)
- Launch the Arduino Open Source Software

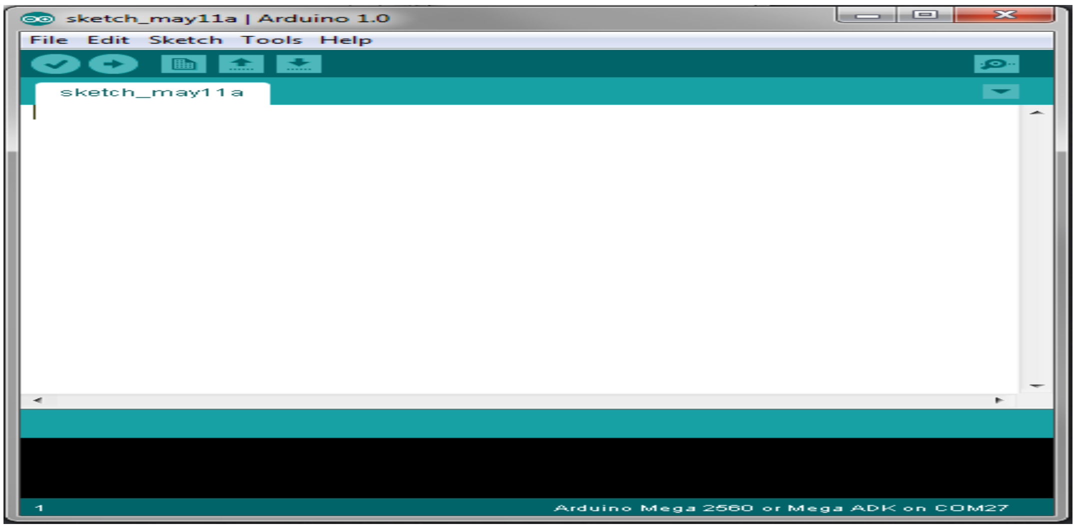
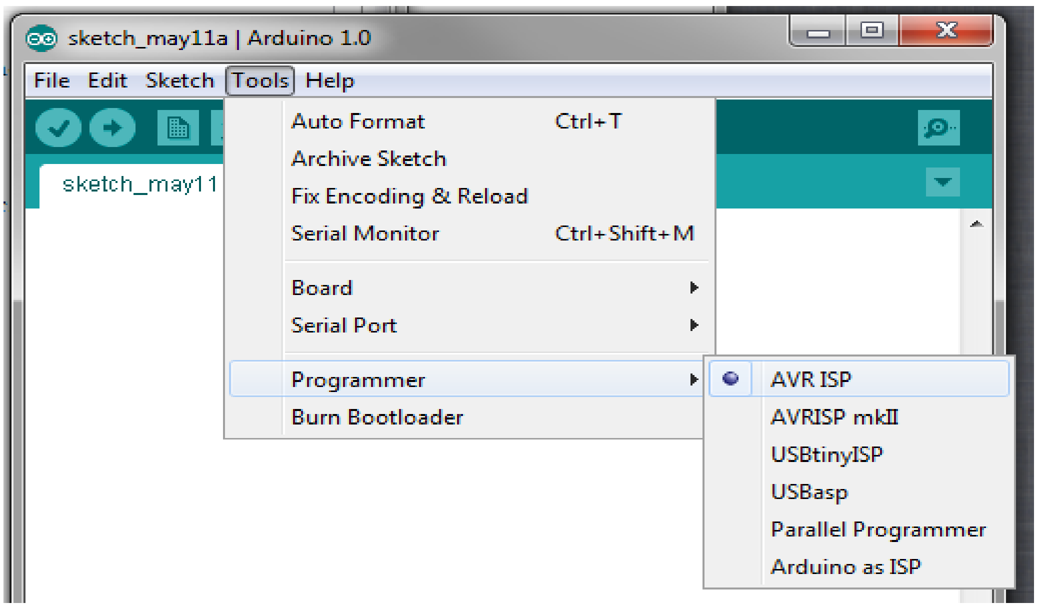
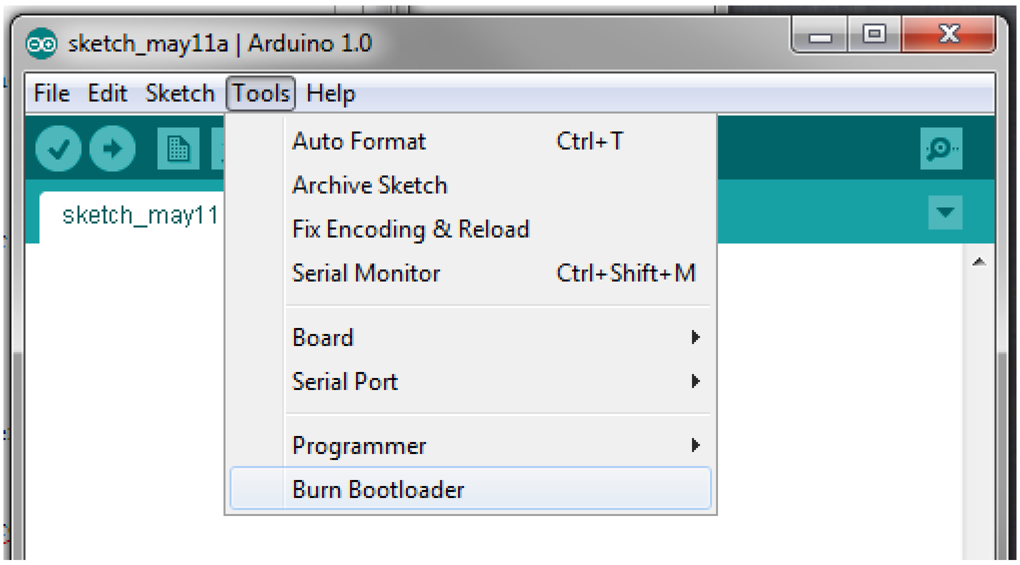

sketch_may11a | Arduino 1.0
File Edit Sketch Tools Help
sketch_may11a
1
Arduino Mega 2560 or Mega ADK on COM27

FIG 1.11

- 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.

5.1.4. Technical Details

The Arduino Boot loader sets the "erase Address" to zero every time the boot loader is called. ROBOTC called the "Load Address" command to set the address in which we want to write/verify when downloading program.

When writing a page of memory to the arduino, the Arduino boot loader will erase the existing page and write a whole new page.

In the scenario of downloading firmware, everything is great because the Erase Address and the Loaded Address both start at zero.

In the scenario of writing a user program, we start writing at memory location 0x7000, but the Boot loader erases information starting at location zero because the "Load Address" command doesn't update where to erase.

Our modification is to set both the Load Address and the Erase Address so the activity of writing a user program doesn't cause the firmware to be accidentally erased.

5.1.5. Summary

Microcontroller	Arduino UNO
Operating Voltage	5 V Input Voltage (recommended)
Input Voltage (limits)	6-20 V
Digital I/O pins	54 (14 provide PWM output)
Analog Input pins	16
DC current per I/O pin	40 mA
DC current for 3.3 V pin	50 mA
Flash Memory	256 KB (8KB used by bootloader)
SRAM	8 KB
EEPROM	4 KB
Lock Speed	16 MHz

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 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the programmed 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** A 3.3V supply generated by the on-board regulator
- **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^{2}C: 20 (SDA) and 21 (SCL).** Support I^{2}C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^{2}C 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.

5.1.6. Communication

The Arduino UNO has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The Arduino UNO provides four hardware UARTs for TTL (5V) serial communication.

An ATMEGA on the board channels one of these over USB and provides a virtual comport to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and1).

A SoftwareSerial library allows for serial communication on any of the digital pins.

The Arduino UNO also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the Arduino UNO datasheet.

5.1.7. Programming

The Arduino UNO can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Arduino UNO on the Arduino UNO comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

EMBEDDED – C PROGRAM :

```
#include<LiquidCrystal.h>

const int rs = 5, en = 18, d4 = 19, d5 = 21, d6 = 22, d7 = 23;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int FLAME11 = 36;

int PUMP = 2;

int ECHOpin = 26;

int TRIGpin = 25;

long duration;

int distance;

int FLAME22 = 34;

int FLAME33 = 35;


int M1 = 32;

int M2 = 33;

int M3 = 13;

int M4 = 12;


void setup()
```

```
{
 pinMode(TRIGpin, OUTPUT);
 pinMode(ECHOpin, INPUT);
 pinMode(PUMP, OUTPUT);
 pinMode(FLAME11, INPUT);
 pinMode(FLAME22, INPUT);
 pinMode(FLAME33, INPUT);
 pinMode(M1,  OUTPUT);
 pinMode(M2,  OUTPUT);
 pinMode(M3,  OUTPUT);
 pinMode(M4, OUTPUT);

 digitalWrite(M1,LOW);
 digitalWrite(M2,LOW);
 digitalWrite(M3,LOW);
 digitalWrite(M4,LOW);

 Serial.begin (9600);
 lcd.begin(16, 2);
 lcd.setCursor(0, 0);
 lcd.print(" FIRE FIGHTING ");
 lcd.setCursor(0, 1);
 lcd.print("      ROBOT      ");
 delay(4000);
 lcd.clear();
}

void loop()
{
 int FLAME1=digitalRead(FLAME11);
 int FLAME2=digitalRead(FLAME22);
 int FLAME3=digitalRead(FLAME33);
 digitalWrite(TRIGpin, LOW);
 delayMicroseconds(4);
 digitalWrite(TRIGpin, HIGH);
 delayMicroseconds(15);
 digitalWrite(TRIGpin,  LOW);
 duration = pulseIn(ECHOpin, HIGH);
 distance = duration*(0.034/2);
 lcd.setCursor(0,0);
 lcd.print("D : ");
 lcd.print(distance);
 lcd.print(" ");
lcd.setCursor(0,1);
lcd.print("F1:");
lcd.print(FLAME1);
lcd.print("     ");

lcd.setCursor(6,1);
lcd.print("F2:");
lcd.print(FLAME2);
lcd.print("     ");

lcd.setCursor(11,1);
lcd.print("F3:");
lcd.print(FLAME3);
lcd.print("     ");



Serial.print("D="); // Print litres/hour
Serial.println(distance);
Serial.print("F="); // Print litres/hour
Serial.println(FLAME1);
```

```
delay(550);


if(FLAME1 == 0)
{
  lcd.setCursor(0,1);
  lcd.print("FIRE DETECTED        ");
  lcd.print("     ");
  digitalWrite(M1,HIGH);
  digitalWrite(M2,LOW);
  digitalWrite(M3,HIGH);
  digitalWrite(M4,LOW);
   delay (4000);
  digitalWrite(M1,LOW);
  digitalWrite(M2,LOW);
  digitalWrite(M3,LOW);
  digitalWrite(M4,LOW);


  digitalWrite(PUMP,HIGH);
  Serial.println("FIRE DETECTED");
  delay(3000);
}
else if (FLAME2 == 0)
```

```
{
 lcd.setCursor(0,1);
 lcd.print("FIRE DETECTED       ");
 lcd.print("     ");

 digitalWrite(M1,HIGH);
 digitalWrite(M2,LOW);
 digitalWrite(M3,LOW);
 digitalWrite(M4,LOW);
 delay (3500);
 digitalWrite(M1,HIGH);
 digitalWrite(M2,LOW);
 digitalWrite(M3,HIGH);
 digitalWrite(M4,LOW);
 delay(2500);
 digitalWrite(M1,LOW);
 digitalWrite(M2,LOW);
 digitalWrite(M3,LOW);
 digitalWrite(M4,LOW);

 digitalWrite(PUMP,HIGH);
 Serial.println("FIRE DETECTED");
```

```
  delay(3000);
}

 else if (FLAME3 == 0)
{
  lcd.setCursor(0,1);
  lcd.print("FIRE DETECTED       ");
  lcd.print("     ");

  digitalWrite(M1,LOW);
  digitalWrite(M2,LOW);
  digitalWrite(M3,HIGH);
  digitalWrite(M4,LOW);
  delay (3500);
  digitalWrite(M1,HIGH);
  digitalWrite(M2,LOW);
  digitalWrite(M3,HIGH);
  digitalWrite(M4,LOW);
  delay(2500);
  digitalWrite(M1,LOW);
  digitalWrite(M2,LOW);
  digitalWrite(M3,LOW);
  digitalWrite(M4,LOW);


  digitalWrite(PUMP,HIGH);
  Serial.println("FIRE DETECTED");
  delay(3000);
}

 else if ((FLAME3 == 0)&&(FLAME1 == 0))
{
  lcd.setCursor(0,1);
  lcd.print("FIRE DETECTED       ");
  lcd.print("     ");
  digitalWrite(M1,HIGH);
  digitalWrite(M2,LOW);
  digitalWrite(M3,HIGH);
  digitalWrite(M4,LOW);
   delay (4000);
  digitalWrite(M1,LOW);
  digitalWrite(M2,LOW);
  digitalWrite(M3,LOW);
  digitalWrite(M4,LOW);
  digitalWrite(PUMP,HIGH);
  Serial.println("FIRE DETECTED");
  delay(3000);
}

   else if ((FLAME2 == 0)&&(FLAME1 == 0))
{
  lcd.setCursor(0,1);
  lcd.print("FIRE DETECTED        ");
  lcd.print("     ");
  digitalWrite(M1,HIGH);
  digitalWrite(M2,LOW);
  digitalWrite(M3,HIGH);
  digitalWrite(M4,LOW);
  delay (4000);
  digitalWrite(M1,LOW);
  digitalWrite(M2,LOW);
  digitalWrite(M3,LOW);
  digitalWrite(M4,LOW);

  digitalWrite(PUMP,HIGH);
  Serial.println("FIRE DETECTED");
  delay(3000);
```

```
}


 else if ((FLAME3 == 0)&&(FLAME2 == 0))
{
  lcd.setCursor(0,1);
  lcd.print("FIRE DETECTED        ");
  lcd.print("     ");
  digitalWrite(M1,HIGH);
  digitalWrite(M2,LOW);
  digitalWrite(M3,LOW);
  digitalWrite(M4,LOW);
  delay (3500);
  digitalWrite(M1,HIGH);
  digitalWrite(M2,LOW);
  digitalWrite(M3,HIGH);
  digitalWrite(M4,LOW);
  delay(2500);
  digitalWrite(M1,LOW);
  digitalWrite(M2,LOW);
  digitalWrite(M3,LOW);
  digitalWrite(M4,LOW);
```

```
      digitalWrite(PUMP,HIGH);
      Serial.println("FIRE DETECTED");
      delay(3000);
    }
    else
    {
      digitalWrite(PUMP,LOW);
      digitalWrite(M1,LOW);
      digitalWrite(M2,LOW);
      digitalWrite(M3,LOW);
      digitalWrite(M4,LOW);
    }
  delay(1000);
  }
```

5.1.8. Automatic Software (Reset)

Rather than requiring a physical press of the reset button before an upload, the Arduino UNO is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the Arduino UNO via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Arduino UNO is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the UNO. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

5.1.9. USB Over Current Protection

5.1.9.1. Physical Characteristics and Shield Compatibility

The Arduino UNO has a resettable poly fuse that protects your computer's USB ports from shorts and over current. Although most computers provide their own internal protection, the fuse provides an extra layer ofprotection.Ifmorethan500mAisappliedtotheUSBport, the fuse will automatically break the connection until the short or overload is removed.

The maximum length and width of the UNO PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The UNO is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila.

Please note that I^2C is not located on the same pins on the Mega (20and21) as the Duemilanove / Diecimila (analog inputs 4 and 5).

5.1.9.2. Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. http://arduino.cc/en/Guide/HomePage
Once you have downloaded/unzipped the arduino IDE, you can plug the Arduino to your PC via USB cable.

5.1.10. Embedded C

High-level language programming has long been in use for embedded-systems development. However, assembly programming still prevails, particularly for digital-signal processor (DSP) based systems. DSPs are often programmed in assembly language by programmers who know the processor architecture inside out. The key motivation for this practice is performance, despite the disadvantages of assembly programming when compared to high-level language programming.

If the video decoding takes 80 percent of the CPU-cycle budget instead of 90 percent, for instance, there are twice as many cycles available for audio processing. This coupling of performance to end-user features is characteristic of many of the real-time applications in which DSP processors are applied. DSPs have a highly specialized architecture to achieve the performance requirements for signal processing applications within the limits of cost and power consumption set for consumer applications. Unlike a conventional Load-Store (RISC) architecture, DSPs have a data path with memory-access units that directly feed into the arithmetic units. Address registers are taken out of the general-purpose register file and placed next to the memory units in a separate register file.

A further specialization of the data path is the coupling of multiplication and addition to form a single cycle Multiply-accumulate unit (MAC). It is combined with special-purpose accumulator registers, which are separate from the general-purpose registers. Data memory is segmented and placed close to the MAC to achieve the high bandwidths required to keep up with the streamlined data path. Limits are often placed on the extent of memory-addressing operations. The localization of resources in the data path saves many data movements that typically take place in a Load-Store architecture.

The most important, common arithmetic extension to DSP architectures is the handling of saturated fixed-point operations by the arithmetic unit. Fixed-point arithmetic can be implemented with little additional cost over integer arithmetic. Automatic saturation (or clipping) significantly reduces the number of control-flow instructions needed for checking overflow explicitly in the program.Changes in technological and economic requirements make it more expensive to continue programming DSPs in assembly. Staying with the mobile phone as an example, the signal-processing algorithms required become increasingly complex. Features such as stronger error correction and encryption must be added. Communication protocols become more sophisticated and require much more code to implement. In certain markets, multiple protocol stacks are implemented to be compatible with multiple service providers. In addition, backward compatibility with older protocols is needed to stay synchronized with provider networks that are in a slow process of upgrading.

Today, most embedded processors are offered with C compilers. Despite this, programming DSPs is still done in assembly for the signal processing parts or, at best, by using assembly-written libraries supplied by manufacturers. The key reason for this is that although the architecture is well matched to the requirements of the signal-processing application, there is no way to express the algorithms efficiently and in a natural way in Standard C. Saturated arithmetic.

For example, is required in many algorithms and is supplied as a primitive in many DSPs. However, there is no such primitive in Standard C. To express saturated arithmetic in C requires comparisons, conditional statements, and correcting assignments. Instead of using a primitive, the operation is spread over a number of statements that are difficult to recognize as a single primitive by a compiler.

5.1.10.1. Description

Embedded C is designed to bridge the performance mismatch between Standard C and the embedded hardware and application architecture. It extends the C language with the primitives that are needed by signal-processing applications and that are commonly provided by DSP processors. The design of the support for fixed-point data types and named address spaces in Embedded C is based on DSP-C. DSP-C [1] is an industry-designed extension of C with which experience was gained since 1998 by various DSP manufacturers in their compilers. For the development of DSP-C by ACE (the company three of us work for), cooperation was sought with embedded-application designers and DSP manufacturers.

The Embedded C specification extends the C language to support freestanding embedded processors in exploiting the multiple address space functionality, user-defined named address spaces, and direct access to processor and I/O registers. These features are common for the small, embedded processors used in most consumer products. The features introduced by Embedded C are fixed-point and saturated arithmetic, segmented memory spaces, and hardware I/O addressing. The description we present here addresses the extensions from a language-design perspective, as opposed to the programmer or processor architecture perspective.

5.1.10.2. Multiple Address Spaces

Embedded C supports the multiple address spaces found in most embedded systems. It provides a formal mechanism for C applications to directly access (or map onto) those individual processor instructions that are designed for optimal memory access. Named address spaces use a single, simple approach to grouping memory locations into functional groups to support MAC buffers in DSP applications, physical separate memory spaces, direct access to processor registers, and user-defined address spaces.

The Embedded C extension supports defining both the natural multiple address space built into a processor's architecture and the application-specific address space that can help define the solution to a problem.

Embedded C uses address space qualifiers to identify specific memory spaces in variable declarations. There are no predefined keywords for this, as the actual memory segmentation is left to the implementation. As an example, assume that **X** and **Y** are memory qualifiers. The definition:

```
X int a[25];
```

Means that **a** is an array of 25 integers, which is located in the **X** memory. Similarly (but less common):

```
X int *Y p;
```

Means that the pointer **p** is stored in the **Y** memory. This pointer points to integer data that is located in the **X** memory. If no memory qualifiers are used, the data is stored into unqualified memory.

For proper integration with the C language, a memory structure is specified, where the unqualified memory encompasses all other memories. All unqualified pointers are pointers into this unqualified memory. The unqualified memory abstraction is needed to keep the compatibility of the **void *** type, the **NULL** pointer, and to avoid duplication of all library code that accesses memory through pointers that are passed as parameters.

5.1.10.3. Named Registers

Embedded C allows direct access to processor registers that are not addressable in any of the machine's address spaces. The processor registers are defined by the compiler-specific, named-register, storage class for each supported processor. The processor registers are declared and used like conventional C variables (in many cases volatile variables). Developers using Embedded C can now develop their applications, including direct access to the condition code register and other processor-specific status flags, in a high-level language, instead of inline assembly code.

Named address spaces and full processor access reduces application dependency on assembly code and shifts the responsibility for computing data types, array and structure offsets, and all those things that C compilers routinely and easily do from developers to compilers.

5.1.10.4. I/O Hardware Address

The motivation to include primitives for I/O hardware addressing in Embedded C is to improve the portability of device-driver code. In principle, a hardware device driver should only be concerned with the device itself. The driver operates on the device through device registers, which are device specific. However, the method to access these registers can be very different on different systems, even though it is the same device that is connected. The I/O hardware access primitives aim to create a layer that abstracts the system-specific access method from the device that is accessed. The ultimate goal is to allow source-code portability of device drivers between different systems. In the design of the I/O hardware-addressing interface, three requirements needed to be fulfilled:

1. The device-drive source code must be portable.
2. The interface must not prevent implementations from producing machine code that is as efficient as other methods.
3. The design should permit encapsulation of the system-dependent access method.

The design is based on a small collection of functions that are specified in the <iohw.h> include file. These interfaces are divided into two groups; one group provides access to the device, and the second group maintains the access method abstraction itself.

To access the device, the following functions are defined by Embedded C:

```
unsigned int iord(  ioreg_designator );
void iowr( ioreg_designator, unsigned int value );
void ioor( ioreg_designator, unsigned int value );
void ioand( ioreg_designator, unsigned int value );
void ioxor( ioreg_designator, unsigned int value );
```

These interfaces provide read/write access to device registers, as well as typical methods for setting/resetting individual bits. Variants of these functions are defined (with **buf** appended to the names) to access arrays of registers. Variants are also defined (with l appended) to operate with **long** values.

All of these interfaces take an I/O register designator **ioreg_designator** as one of the arguments. These register designators are an abstraction of the real registers provided by the system implementation and hide the access method from the driver source code. Three functions are defined for managing the I/O register designators. Although these are abstract entities for the device driver, the driver does have the obligation to initialize and release the access methods. These functions do not access or initialize the device itself because that is the task of the driver. They allow, for example, the operating system to provide a memory mapping of the device in the user address space.

```
void iogroup_acquire( iogrp_designator );
void iogroup_release( iogrp_designator );
void iogroup_map( iogrp_designator, iogrp_designator );
```

The **iogrp_designator** specifies a logical group of I/O register designators; typically this will be all the registers of one device. Like the I/O register designator, the I/O group designator is an identifier or macro that is provided by the system implementation. The map variant allows cloning of an access method when one device driver is to be used to access multiple identical devices.

5.1.10.5. Embedded C Portability

By design, a number of properties in Embedded C are left implementation defined. This implies that the portability of Embedded C programs is not always guaranteed. Embedded C provides access to the performance features of DSPs. As not all processors are equal, not all Embedded C implementations can be equal For example, suppose an application requires 24-bit fixed-point arithmetic and an Embedded C implementation provides only 16 bits because that is the native size of the processor. When the algorithm is expressed in Embedded C, it will not produce outputs of the right precision.

In such a case, there is a mismatch between the requirements of the application and the capabilities of the processor. Under no circumstances, including the use of assembly, will the algorithm run efficiently on such a processor. Embedded C cannot overcome such discrepancies. Yet, Embedded C provides a great improvement in the portability and software engineering of embedded applications. Despite many differences between performance-specific processors, there is a remarkable similarity in the special-purpose features that they provide to speed up applications.

Writing C code with the low-level processor-specific support may at first appear to have many of the portability problems usually associated with assembly code. In the limited experience with porting applications that use Embedded C extensions, an automotive engine controller application (about 8000 lines of source) was ported from the eTPU, a 24-bit special-purpose processor, to a general-purpose 8-bit Freescale 68S08 with about a screen full of definitions put into a single header file. The porting process was much easier than expected. For example, variables that had been implemented on the processor registers were ported to unqualified memory in the general-purpose microprocessor by changing the definitions in the header definition and without any actual code modifications. The exercise was to identify the porting issues and it is clear that the performance of the special-purpose processor is significantly higher than the general-purpose target.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Fire detection is a serious issue. They provide us greater efficiency to detect the flame and it can be extinguish before it become uncontrollable and threat to life. Hence, this robot can play a crucial role. Fire fighting robot can be easily and conveniently used. Operate automatically when any fire occurs. Robot comprises of very small size, less in weight, hence require less space.

In this study, a genetic algorithm based fire search algorithm and a vision based fire front line tracking and monitoring algorithm are proposed. These two methods have been proved to work effectively for forest fires searching and monitoring through simulations. Multiple UAVs working together within a cooperative strategy presents a practical and powerful tool for forest fires monitoring, data gathering as well as fire fighting. Such a solution provides a flexible way to deal with forest fire disasters. Despite the fact that the proposed approach is promising, there still remains a few challenges. For example, fuel consumption and refuelling, irregular fire shapes and coordination with other fire fighting devices are still open problems.

REFERENCES

[1] R. W. Davis, "Bio-logging as a Method for Understanding Natural Systems," International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008), pp. 12-17, 2008.

[2] Liyang Yu, et al., "Real-time forest fire detection with wireless sensor networks," Proc. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1214-1217, 2005.

[3] K. Tanaka, et al.,"Design of Operating Software and Electrical System of Mobile Robot for Environmental Monitoring," proceedings of 2014 IEEE International Conference on Robotics and Biomimetics, 2014.

[4] S. A. Restrepo, et al., "Path planning applied to the mobile robot GBOT", Proc. 2012 XVII Symposium of Image and Artificial Vision (STSIVA), pp. 281-288, 2012.

[5] D. Amorim, et al., "A physics-based optimization approach for path planning on rough terrains," Proc. 12th International conference on informatics in control, Automation and robotics (ICINCO), pp. 259-266, 2015.

[6] T. Ohki, et al., "Path planning for mobile robot on rough terrain based on sparse transition cost propagation in extended elevation maps", Proc. 2013 IEEE International conference on mechatoronics and automation, pp.495-499, 2013.

[7] K. Tanaka et at., “A Study of a Wheel Shape for Increasing Climbing Ability of Slopes and Steps”, in Proc. 21st CISM IFToMM Symposium on Robot Design, Dynamics and Control (ROMANSY 2016), Jun 2016.

[8] Brooks, et al.:"A robust layered control system for a mobile robot.", IEEE Journal of 2.1: 14-23, 1986.

[9] Payton, et al., “An architecture for reflexive autonomous vehicle control,” Proc. 1986 IEEE International Conference on Robotics and Automation. Vol. 3, pp. 1838 - 1845, 1986.

[10] Connell, et al.,”SSS: A hybrid architecture applied to robot navigation,” Proc. 1992 IEEE International Conference on Robotics and Automation, Vol. 3, pp. 2719 – 2724, 1992.

[11] Kupfermann, Irving. “Turn alternation in the pill bug (Armadillidium vulgare).” Animal behaviour 14.1 pp. 68-72, 1966.

[12] K. Tanaka, et al, “Hardware and control design considerations for a monitoring system of autonomous mobile robots in extreme enviroment”, Proc. 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2017.

[13] K. Tanaka, et al., “A Novel Approach to Increase the Locomotion Performance of Mobile Robots in Fields With Tall Grasses”, in IEEE Robotics and Automation Letters, vol. 1, no. 1, pp. 122-129, 2016.

[14] K. Tanaka et al., "Novel extendable arm structure using convex tapes for improving strength of pipe on tiny mobile robots," Proc. 2016 IEEE International Conference on Robotics and Biomimetics, pp. 637-642, 2016.

[15] C. Yuan, Y. M. Zhang, and Z. Liu, “A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques,” Canadian Journal of Forest Research, vol. 45, no. 7, pp.783–792,2015.