

Gps Enabled Vehicle Location Identification Using Gsm And Fare Collection Using Smart Card

^Ak.G.Revathi, ^Bdr. Belsam Jeba Ananth, ^Cm , L.Saravanan, ^Da.Ranjith Kumar

^aProfessor / Electronics and Communication Engineering

DMI COLLEGE OF ENGINEERING

revathi30pranav@gmail.com

^bAssociate professor

Department of Mechatronics engineering

SRM Institute of Science and technology, Kattankulathur, Chengalpattu

belsamjebaananth@gmail.com

^cAssistant Professor

Department Of ECE,

Rajalakshmi Institute of Technology,

Chennai, Tamilnadu, India

saravanan.l@ritchennai.edu.in

^dAssistant Professor

Department of Computer Science and Engineering,

Lovely Professional University, Phagwara, India.

ranjithdr.kumar@gmail.com

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: This project explains that the prediction of bus arrival time and bus location information to the passengers in their respective bus stops through GSM and GPS and by measuring the speed by accelerometer. It includes that the unmanned vehicle(i.e. bus) driver and passengers journeying without conductor. It also include an automated fare collection system through passengers onboard smart card tickets. Our project provides vehicle location updated information to the passengers and other additional information's are, number of passengers onboard while starting the bus in the starting point and how many of them off board and onboard in point to point stopping information and also provides final passengers information through GSM to ticket checker and other higher officials and also an information viewed by driver through electronic LCD board.

I. Introduction

While travelling with buses, the travelers initially want to know the accurate arrival time of the bus. In our proposed system, we present a innovative bus arrival time prediction system based on accelerometer and sensors. Our embedded system field provides onboard off board passengers ticket information through GSM to other higher officials. In the previous years, onboard/ off board in point to point stopping information through GSM in ever implemented. Finally, our embedded system provides a module such as GPS, GSM, Arduino ATmega microcontroller, etc... to execute and implement this project in embedded field.

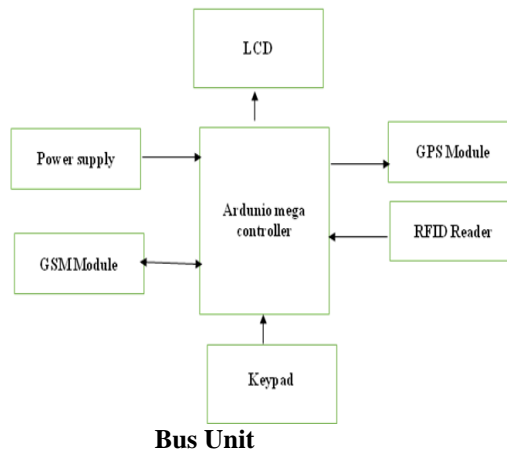
OBJECTIVE:

To avoid long wait of a bus passengers in their respective stations. To avoid ticket checking and providing in an onboard bus vehicles for the transport uses.

PROPOSED SYSTEM:

- Automatic fare collection system to avoid the man power work and reducing money expenditure from government fund.
- No need to consider the onboard without ticket passengers.
- Updated predicting bus arrival time through GPS and Accelerometer.
- Onboard/off board passengers ticket information through GSM to ticket checker or other officials and also this information goes to driver through electronic LCD display in the bus.

**BLOCK DIAGRAM:
Station Unit**



Bus Unit

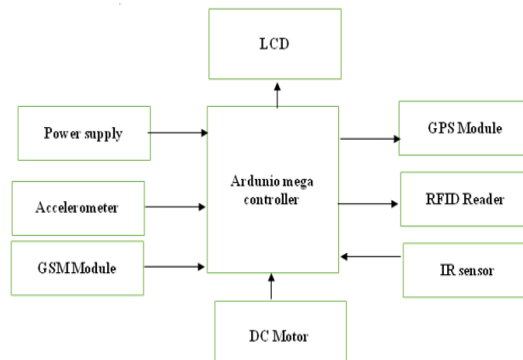


Fig1.1 Block Diagram Bus and station unit

HARDWARE REQUIREMENTS:

Power Supply
 Arduino mega Controller
 GPS Module
 GSM Module
 IR Sensors
 RFID Reader
 Accelerometer
 DC Motor
 DC Driver
 Keypad
 LCD Display

SOFTWARE REQUIREMENTS:

Arduino 'C' compiler
 Embedded C

ADVANTAGES

- Real time updating.
- Necessity of conductor and ticket checker is avoidable.
- Alert message to the sleeping passengers.
- Arrival time and onboard passengers identification through GSM and accelerometer.
- Bus or train location identification through GPS whenever the vehicle is congested.

DRIVER MODULE

POWER SUPPLY

There are many types of power supply. Most are designed to convert high voltage AC mains electricity to a suitable low voltage supply for electronics' circuits and other devices. A power supply can be broken down into a series of blocks, each of which performs a particular function.

For example a 5V regulated supply:

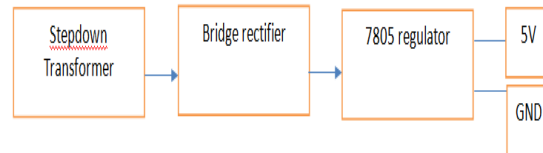


Fig V regulated supply

Step Down Transformers

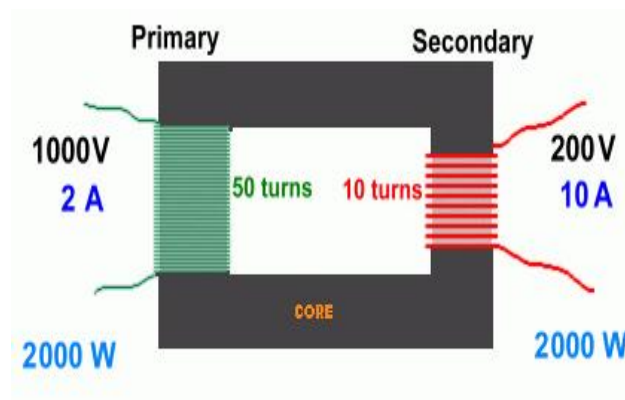


Fig Step Down Transformer

Step down transformers are designed to reduce electrical voltage. Their primary voltage is greater than their secondary voltage. This kind of transformer "steps down" the voltage applied to it. For instance, a step down transformer is needed to use a 110v product in a country with a 220v supply.

Step down transformers convert electrical voltage from one level or phase configuration usually down to a lower level. They can include features for electrical isolation, power distribution, and control and instrumentation applications. Step down transformers typically rely on the principle of magnetic induction between coils to convert voltage and/or current levels.

Step down transformers are made from two or more coils of insulated wire wound around a core made of iron. When voltage is applied to one coil (frequently called the primary or input) it magnetizes the iron core, which induces a voltage in the other coil, (frequently called the secondary or output). The turns ratio of the two sets of windings determines the amount of voltage transformation.

An example of this would be: 100 turns on the primary and 50 turns on the secondary, a ratio of 2 to 1. Step down transformers can be considered nothing more than a voltage ratio device.

With step down transformers the voltage ratio between primary and secondary will mirror the "turns ratio" (except for single phase smaller than 1 kva which have compensated secondaries). A practical application of this 2 to 1 turns ratio would be a 480 to 240 voltage step down. Note that if the input were 440 volts then the output would be 220 volts. The ratio between input and output voltage will stay constant. Transformers should not be operated at voltages higher than the nameplate rating, but may be operated at lower voltages than rated. Because of this it is possible to do some non-standard applications using standard transformers.

Single phase step down transformers 1 kva and larger may also be reverse connected to step-down or step-up voltages. (Note: single phase step up or step down transformers sized less than 1 KVA should not be reverse connected because the secondary windings have additional turns to overcome a voltage drop when the load is applied. If reverse connected, the output voltage will be less than desired.)

Bridge rectifier

A bridge rectifier is an arrangement of four or more diodes in a bridge circuit configuration which provides the same output polarity for either input polarity. It is used for converting an alternating current (AC) input into a direct current (DC) output. A bridge rectifier provides full-wave rectification from a two-wire AC input, therefore resulting in lower weight and cost when compared to a rectifier with a 3-wire input from a transformer with a center-tapped secondary winding.

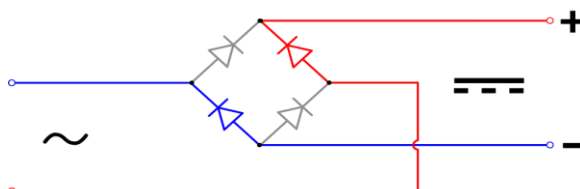


Fig Bridge rectifier

Types of Bridge Rectifiers

There are many different kinds of bridge rectifiers and at Future Electronics we stock many of the most common types categorized by maximum average rectified current, maximum reverse voltage, maximum peak current, forward voltage, packaging type and maximum reverse current. The parametric filters on our website can help refine your search results depending on the required specifications.

The most common sizes for maximum average rectified current are 1A, 1.5 A, 4 A, 25 A and 35 A. We also carry bridge rectifiers with maximum average rectified current as high as 1000 A. Forward voltage can range from 450 mV to 1.1 kV, with the most common bridge rectifier semiconductor chips having a forward voltage of 1.1 V or 1 V.

Voltage Regulator

Voltage regulator IC's are the IC's that are used to regulate voltage.

IC 7805 is a 5V Voltage Regulator that restricts the voltage output to 5V and draws 5V regulated power supply. It comes with provision to add heat sink.

The maximum value for input to the voltage regulator is 35V. It can provide a constant steady voltage flow of 5V for higher voltage input till the threshold limit of 35V. If the voltage is near to 7.5V then it does not produce any heat and hence no need for heatsink. If the voltage input is more, then excess electricity is liberated as heat from 7805.

It regulates a steady output of 5V if the input voltage is in range of 7.2V to 35V. Hence to avoid power loss try to maintain the input to 7.2V. In some circuitry voltage fluctuation is fatal (for e.g. Microcontroller), for such situation to ensure constant voltage IC 7805 Voltage Regulator is used. For more information on specifications of 7805 Voltage Regulator please refer the data sheet here ([IC 7805 Voltage Regulator Data Sheet](#)).

IC 7805 is a series of 78XX voltage regulators. It's a standard, from the name the last two digits 05 denotes the amount of voltage that it regulates. Hence a 7805 would regulate 5v and 7806 would regulate 6V and so on.

The schematic given below shows how to use a 7805 IC, there are 3 pins in IC 7805, pin 1 takes the input voltage and pin 3 produces the output voltage. The GND of both input and out are given to pin 2.

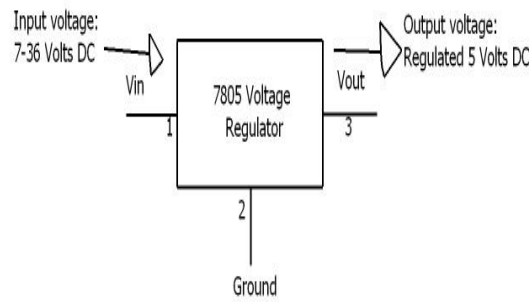


Fig 7805 Voltage Regulator Ic Circuit

Circuit Diagram:

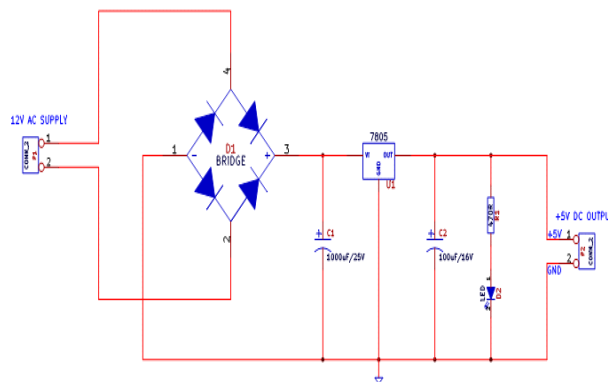


Fig power supply Circuit Description:

The 230/240VAC line voltage is applied to the transformer primary. Step down transformer is used to convert 230VAC to 12V/1A output voltage in the transformer secondary. This 12VAC supply is applied to the bridge rectifier. The bridge rectifier voltage rating should be double the Vrms of secondary AC and higher forward current rating. The bridge rectifier has four diodes in that, available as singular. This is the most important and it produces full-wave varying DC or pulsating DC. This rectified output is smoothed by using shunt capacitor filter (C1). The larger the filter capacitor lowers the ripple. The larger capacitor C1 across the input bypasses AC ripples to ground. The pure DC output reaches the regulator IC. The 7805 Voltage regulator IC (U1) gives 5V /1A regulated output. The capacitor (C2) across the output improves transient response. Low power red LED is used in the output for power indication. It has voltage drop of 1.8V and gives brightness at 20 Ma. This is done by current limiting resistor (R1).

$$R = (V_{out} - V_F) / I$$

Where, Vout –output voltage; VF- led voltage drop; I- led forward current.

From this we get R as 160E. We use the standard value of 220E. The higher value of limiting resistor gives low brightness but longer life time. So, here 470E is used.

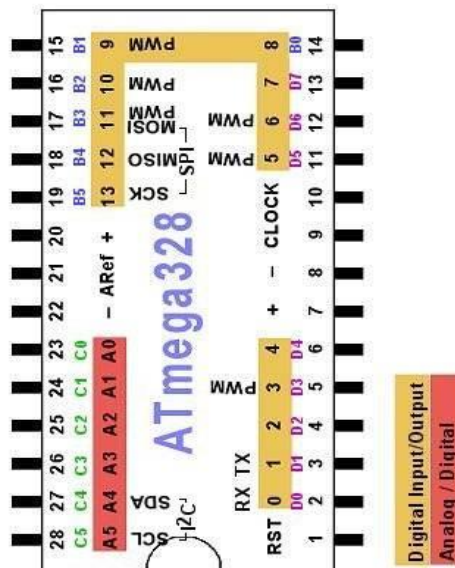
II ARDUINO MICROCONTROLLER

ATMEGA 328:

Arduino is common term for a software company, project, and user community, that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices supports the languages C and C++.The first Arduino was introduced in 2005.

The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers.

Microcontroller:



Port B has pins **B0 to B5**
 Port C has pins **C0 to C5**
 Port D has Pins **D0 to D7**

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega48PA/88PA/168PA/328P provides the following features: 4/8/16/32K bytes of In- System Programmable Flash with Read-While-Write capabilities, 256/512/512/1K bytes EEPROM, 512/1K/1K/2K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The ATmega48PA/88PA/168PA/328P AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

Comparison Between ATmega48PA, ATmega88PA, ATmega168PA and ATmega328P

The ATmega48PA, ATmega88PA, ATmega168PA and ATmega328P differ only in memory sizes, boot loader support, and interrupt vector sizes.

Device	Flash	EEPRO	RAM	InterruptVector
ATmega4	4K	256	512	1instruction
ATmega8	8K	512	1K	1instruction
ATmega1	16K	512	1K	2instruction
ATmega3	32K	1K	2K	2instruction

Table.Memory Size Summary

ATmega88PA, ATmega168PA and ATmega328P support a real Read-While-Write Self-Programming mechanism. There is a separate Boot Loader Section, and the SPM instruction can only execute from there. In

ATmega48PA, there is no Read-While-Write support and no separate Boot Loader Section. The SPM instruction can execute from the entire Flash.

ARDUINO BOARD:

An Arduino board historically consists of an [Atmel](#) 8-, 16- or 32-bit AVR [microcontroller](#) (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed *shields*. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an [I²C serial bus](#)—so many shields can be stacked and used in parallel. Before 2015, Official Arduinos had used the Atmel [megaAVR](#) series of chips, specifically the [ATmega8](#), [ATmega168](#), [ATmega328](#), [ATmega1280](#), and [ATmega2560](#). In 2015, units by other producers were added. A handful of other processors have also been used by Arduino compatible devices. Most boards include a 5 V [linear regulator](#) and a 16 MHz [crystal oscillator](#) (or [ceramic resonator](#) in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a [boot loader](#) that simplifies uploading of programs to the on-chip [flash memory](#), compared with other devices that typically need an external [programmer](#). This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, boot loader is the default boot loader installed on Arduino UNO.



Fig Arduino Atmega328 Board

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

III ARDUINO C COMPILER:

Arduino with usb

In this tutorial, we assume you're using an [Arduino or Genuino Uno](#) or an [Arduino or Genuino Mega 2560](#). If you are using a retired board as [Arduino Duemilanove](#), [Nano](#) or [Diecimila](#) please refer to the driver installation instructions end of this document. If you have another board, read the corresponding page linked in the main getting started page.

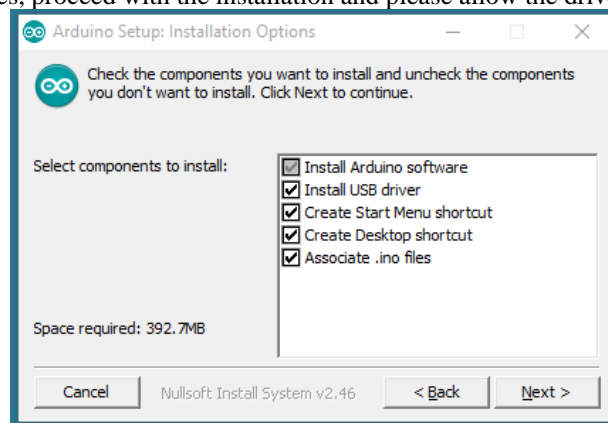
You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example.



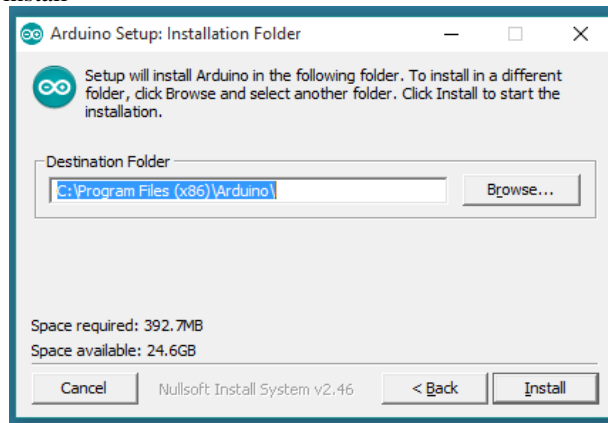
Fig Download and install the Software

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.

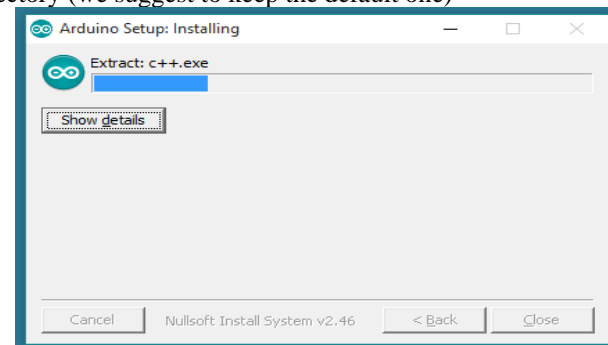
When the download finishes, proceed with the installation and please allow the driver installation process.



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)

Connect the board

The USB connection with the PC is necessary to program the board and not just to power it up. The Uno and Mega automatically draw power from either the USB or an external power supply. Connect the board to your computer using the USB cable. The green power LED (labelled **PWR**) should go on.

Install the board drivers

If you used the Installer, Windows - from XP up to 10 - will install drivers automatically as soon as you connect your board.

If you downloaded and expanded the Zip package or, for some reason, the board wasn't properly recognized, please follow the procedure below.

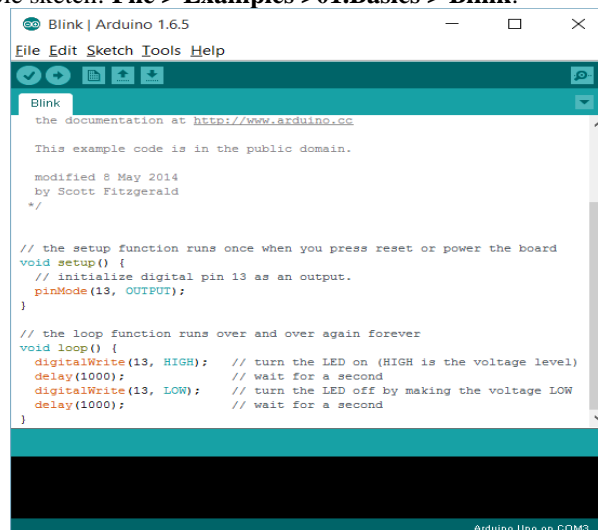
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".
- Right click on the "Arduino UNO (COMxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the driver file named "**arduino.inf**", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named "**Arduino UNO.inf**".
- Windows will finish up the driver installation from there.

Launch the Arduino Software

Double-click the Arduino icon (arduino.exe) created by the installation process. (Note: if the Arduino Software loads in the wrong language, you can change it in the preferences dialog. See [the Arduino Software \(IDE\) page](#) for details.)

Open the blink

Open the LED blink example sketch: **File > Examples > 01.Basics > Blink**.

A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.6.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The main text area contains the following code:

```
Blink
the documentation at http://www.arduino.cc
This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald
*/

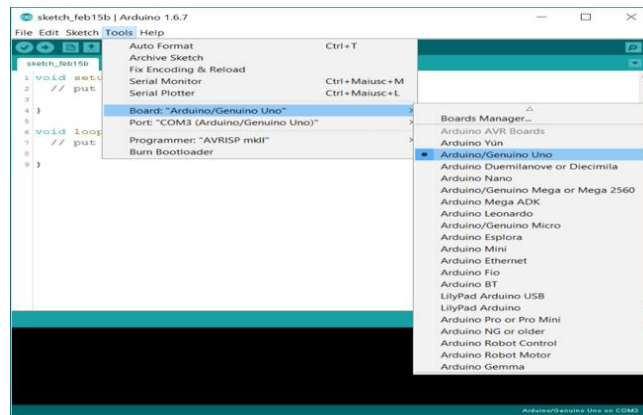
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
```

The status bar at the bottom right indicates "Arduino Uno on COM3".

Select your board

You'll need to select the entry in the **Tools > Board** menu that corresponds to your Arduino or Genuino board.



Selecting an Arduino/Genuino Uno

Details of the board menu entries are available [on the environment page](#).

Select your serial port

Select the serial device of the board from the Tools | Serial Port menu. This is likely to be **COM3** or higher (**COM1** and **COM2** are usually reserved for hardware serial ports). To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino or Genuino board. Reconnect the board and select that serial port.

Upload the program

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.



A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it does, congratulations! You've gotten Arduino or Genuino up-and-running. If you have problems, please see the [troubleshooting suggestions](#).

IV EMBEDDED C

Embedded System Programming

Embedded systems programming is different from developing applications on a desktop computers. Key characteristics of an embedded system, when compared to PCs, are as follows:

- Embedded devices have resource constraints (limited ROM, limited RAM, limited stack space, less processing power)

Components used in embedded system and PCs are different; embedded systems typically uses smaller, less power consuming components. Embedded systems are more tied to the hardware.

Two salient **features of Embedded Programming** are code speed and code size. Code speed is governed by the processing power, timing constraints, whereas code size is governed by available program memory and use of programming language. Goal of embedded system programming is to get maximum features in minimum space and minimum time.

Embedded systems are programmed using different type of languages:

- Machine Code
- Low level language, i.e., assembly
- High level language like C, C++, Java, Ada, etc.
- Application level language like Visual Basic, scripts, Access, etc.

Assembly language maps mnemonic words with the binary machine codes that the processor uses to code the instructions. Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly

difficult to manage in assembly language. Finding good assembly programmers has also become difficult nowadays. Hence high level languages are preferred for embedded systems programming.

Use of **C in embedded systems** is driven by following advantages:

- It is small and reasonably simpler to learn, understand, program and debug.
- C Compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
- Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
- As C combines functionality of assembly language and features of high level languages, C is treated as a 'middle-level computer language' or 'high level assembly language'
- It is fairly efficient
- It supports access to I/O and provides ease of management of large embedded projects.

C AND EMBEDDED C

Though **C and embedded C** appear different and are used in different contexts, they have more similarities than the differences. Most of the constructs are same; the difference lies in their applications.

C is used for desktop computers, while **embedded C** is for microcontroller based applications. Accordingly, C has the luxury to use resources of a desktop PC like memory, OS, etc. While programming on desktop systems, we need not bother about memory. However, embedded C has to use with the limited resources (RAM, ROM, I/Os) on an embedded processor. Thus, program code must fit into the available program memory. If code exceeds the limit, the system is likely to crash.

Compilers for C (ANSI C) typically generate OS dependant executables. **Embedded C** requires compilers to create files to be downloaded to the microcontrollers/microprocessors where it needs to run. Embedded compilers give access to all resources which is not provided in compilers for desktop computer applications.

Embedded systems often have the real-time constraints, which is usually not there with desktop computer applications.

Embedded systems often do not have a console, which is available in case of desktop applications.

So, what basically is different while programming with **embedded C** is the mindset; for embedded applications, we need to optimally use the resources, make the program code efficient, and satisfy real time constraints, if any. All this is done using the basic constructs, syntaxes, and function libraries of 'C'.

V CONCLUSION

This project described a methodology for estimating the destination time of passenger's journey by using GSM. We conclude that a manual work happens in the vehicle during the journey period and also provides final passenger information through GSM to higher officials. Embedded system suited for our proposed system. This project can be developed further as installing navigation map facilities to the passengers in guiding them for finding the difficult routes.

VI References

- [1] T. Abdelzaher et al., "Mobiscopes for human spaces," IEEE Pervasive Compute., vol. 6, no. 2, pp. 20–29, Sep 2015.
- [2] G. Ananthanarayanan, M. Haridasan, I. Mohamed, D. Terry, and C. A. Thekkath, "Startrack: A framework for enabling track-based applications," in Proc. ACM MobiSys, 2014, pp. 207–220.
- [3] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in Proc. IEEE INFOCOM, 2013, pp. 775–784.
- [4] R. K. Balan, K. X. Nguyen, and L. Jiang, "Real-time trip information service for a large taxi fleet," in Proc. ACM MobiSys, 2011, pp. 99–112.
- [5] W. Wang, J. P. Attanucci, and N. H. M. Wilson, "Bus passenger origin destination estimation and related analyses using automated data collection systems," J. Public Transp., vol. 14, no. 4, pp. 131–150, 2011.
- [6] elim s.awad "Voice technology in the instrumentation of the automobile", member IEEE transactions on instrumentation and measurement, VOL. 37, NO. 4. DECEMBER 1988
- [7] Margaret Ducusinha "Modeling of a Series Hybrid Electric High-Mobility Multipurpose Wheeled Vehicle in the journal IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 56, NO. 2, MARCH 2007.
- [8] Ananda ManiPaudel and Philipp Kreutzmann "Design and performance analysis of a hybrid solar tricycle for a sustainable local commute", volume 41, pp.473-482, 2014
- [9] Selvaganapathy Manoharan, Nishavithri Natarajan, "Brain controlled wheelchair for the physically challenged people using Neuro – Sky Sensor", International Journal of Innovative Research in Science, Engineering & Technology, Volume 4, Issue 12, December 2015, P. No.11985 – 11992.

-
- [10] Ravikumar Kandasamy, Sachin Raut, Deep Varma, Ganesh There, "Design of Solar Tricycle for Handicapped Person", volume 5, issue 2, pp.11-24, 2013.
- [11] Lucas H.V. van der Woude, Sonja de Groot and Thomas W.J. Janssen, "Manual wheelchairs: Research and innovation in rehabilitation, sports, daily life and health", Medical Engineering & Physics, volume 28, pp. 905–915, December 2005.
- [12] M. Selvaganapathy, N. Nishavithri, T. Manochandar, G. Manikannan, "Modern vehicle for the physically challenged people using blue eye technology", International Journal of Mechanical Engineering and Technology, 8 (1), 2017, pp. 208 – 212.
- [13] Po Er Hsu, Yeh Liang Hsu, Kai Wei Chang and Claudius Geiser "Mobility assistance design of the Intelligent Robotic Wheelchair", International Journal of Advanced Robotic Systems, volume 9, pp. 1-10, 2012.
- [14] Giuseppe Quaglia, Walter Franco and Riccardo Oderio "Wheelchair.q, a motorized wheelchair with stair climbing ability", Mechanism and Machine Theory, volume 46, pp.1601-1608, 2011.
- [15] Mohd Razali Md Tomaria, Yoshinori Kobayashia, Yoshinori Kunoa, "Development of Smart wheelchair system for a user with severe motor impairment", International Symposium on Robotics and Intelligent Sensors, volume 41, pp. 538-546, 2012.
- [16] P.K. Nag, J.T. Panikar, M.G. Malvankar, C.K. Pradhan and S.K. Chatterjee, "Performance evaluation of lower extremity disabled people with reference to handcranked tricycle propulsion", Applied Ergonomics, volume 13.3, pp. 171-176, 1982.
- [17] Balaji K, R.Nithya, S.Ganesan N.Nishavithri & S.Sakthivel Murugan "modelling Of Submerged Optical Remote Correspondences With Low Losses" International Journal of Mechanical and Production Engineering Research and Development (IJMPERD) Vol. 10, Issue 3, Jun 2020, 3983–3998
- [18] R.Nithya, N.Nishavithri, B.Dheepa, K.Balaji, "Heterogeneous Smart Antenna In Multi-Hop Wireless Network Using Mac Protocol Journal Of Critical Reviews" ISSN- 2394-5125 VOL 7, ISSUE 16, 2020
- [19] Madarasz R.L., Heiny LC., Crompt R.F. and Mazur N.M. (1986). "The design of an Autonomous Vehicle for the Disabled." IEEE Journal of Robotics and Automation, vol. RA2.