Smart Rider Safety Jacket with Emergency Response System

Abstract

This project presents an intelligent rider safety jacket integrated with multiple emergency detection and response systems using STM32 microcontroller. The system incorporates an alcohol sensor that detects rider intoxication and activates a dedicated red LED warning. Fire detection capability triggers red LED illumination when flames are identified, providing immediate visual alerts. An MPU6050 accelerometer and gyroscope module monitors rider movements to detect accidents or falls, triggering comprehensive emergency response protocols. Upon accident detection, the system activates a relay to inflate a protective airbag using a small pump and simultaneously sends emergency SMS alerts with precise GPS coordinates obtained from the NEO-6M GPS module via the SIM800L GSM module. The entire system is powered by a 2-cell Li-ion battery, ensuring portable and reliable operation. This smart jacket provides comprehensive rider protection through real-time hazard monitoring and automated emergency response mechanisms.

Introduction

Motorcycle rider safety remains a critical concern worldwide, with accidents often resulting in severe injuries due to lack of immediate emergency response and protective systems. Traditional safety measures focus on passive protection, lacking intelligent monitoring and automated emergency response capabilities. The increasing number of road accidents involving riders necessitates advanced safety systems that can detect emergencies in real-time and initiate immediate protective actions while alerting emergency services.

This project addresses these safety challenges by developing a smart rider jacket that integrates multiple hazard detection systems with automated emergency response mechanisms. The STM32 microcontroller, programmed through Arduino IDE, serves as the central processing unit that coordinates all safety functions. The alcohol detection system prevents intoxicated riding by providing immediate visual warnings, while the fire detection capability offers protection against fire hazards that riders may encounter.

The innovative accident detection system using MPU6050 continuously monitors rider orientation and movement patterns, accurately identifying falls or collisions. When an accident is detected, the system initiates a dual-response protocol: activating an airbag inflation system for physical protection and sending GPS-coordinated emergency alerts to predefined contacts. The integration of NEO-6M GPS and SIM800L GSM modules ensures that emergency services can quickly locate and assist the rider, significantly reducing response times in critical situations.

Objectives

- To design and implement a smart safety jacket using STM32 microcontroller
- To integrate alcohol detection with visual warning system using red LED
- To implement fire detection capability with immediate alert indication
- To develop accident detection algorithms using MPU6050 sensor
- To create automated airbag inflation system using relay-controlled pump
- To establish emergency SMS alert system with GPS coordinates
- To ensure reliable portable operation using 2-cell Li-ion battery
- To provide comprehensive rider protection through multiple safety layers

Methodology

System Architecture
The project implements a multi-layered safety approach:
- Hazard Detection Layer: Alcohol sensor, fire sensor, MPU6050 motion sensor
- Processing Unit: STM32 microcontroller with emergency decision algorithms
- Response System: LED warnings, airbag inflation, emergency notifications
- Communication Module: GPS location tracking and GSM emergency messaging
- Power Management: 2-cell Li-ion battery for mobile operation

Hazard Detection Systems
Comprehensive threat monitoring implementation:

Alcohol Detection:
- MQ-3 alcohol sensor for breath alcohol concentration monitoring
- Threshold-based detection for intoxication levels
- Dedicated red LED activation when alcohol is detected
- Calibration for environmental compensation

Fire Detection:
- Flame sensor or thermal sensor implementation
- Rapid response to fire hazards near the rider
- Visual warning through red LED illumination
- Multiple sensitivity levels for different environments

Accident Detection:
- MPU6050 accelerometer and gyroscope for motion analysis
- Fall detection algorithms using orientation and impact data
- Sudden deceleration and unusual orientation pattern recognition
- False positive filtering through multi-parameter validation

Emergency Response Protocol
Automated protection and alert system:

Physical Protection:
- Relay-controlled DC pump for airbag inflation
- Rapid inflation mechanism for immediate protection
- Compact airbag system integrated into jacket
- Manual reset and testing capability

Emergency Communication:
- NEO-6M GPS for precise location coordinates
- SIM800L GSM for immediate SMS transmission
- Pre-programmed emergency contacts notification
- Location and situation details in alert messages

Advantages

- Comprehensive Safety: Multiple hazard detection and protection systems
- Immediate Response: Automated emergency protocols without human intervention
- Location Tracking: GPS-enabled emergency service coordination
- Physical Protection: Airbag system reduces impact injuries
- Visual Warnings: LED alerts for various hazard conditions
- Portable Design: Integrated system with battery power
- Cost-effective: Affordable safety enhancement for riders
- Reliable Operation: Robust algorithms for accurate detection
- Life-saving Potential: Significantly improves rider survival chances

Hardware Requirements

- STM32 Microcontroller Development Board
- MQ-3 Alcohol Sensor Module
- Flame Detection Sensor
- MPU6050 Accelerometer & Gyroscope Module
- NEO-6M GPS Module with Antenna
- SIM800L GSM Module with SIM Card
- 5V Relay Module
- DC Air Pump for Inflation
- Inflatable Airbag/Cushion
- Red LEDs (x3 for different alerts)
- 2-Cell Li-ion Battery (18650, 2x) with holder
- Voltage Regulator Circuit
- Jacket with component integration pockets
- Switches and Control Buttons
- Protective Enclosures for electronics

Software Requirements

- Arduino IDE with STM32 Support
- C++ Programming for Embedded Control
- MPU6050 Sensor Libraries and Algorithms
- GPS Data Parsing Libraries
- GSM Communication Protocols
- Accident Detection Algorithms
- Sensor Calibration Routines
- Emergency Response Logic

Implementation Steps

1. Jacket Modification: Create compartments for electronics and airbag
2. Circuit Design: Develop schematics for all sensor integrations
3. Sensor Calibration: Calibrate alcohol, fire, and motion sensors
4. STM32 Programming: Implement hazard detection algorithms
5. Accident Algorithm: Develop MPU6050-based fall detection
6. GPS Integration: Configure location tracking with NEO-6M
7. GSM Setup: Program SIM800L for emergency SMS alerts
8. Airbag System: Integrate pump and inflation mechanism
9. Power Management: Implement 2-cell Li-ion battery system
10. LED Indicators: Wire and program warning light system
11. System Integration: Combine all components in jacket
12. Testing and Validation: Verify all safety functions
13. Field Testing: Conduct real-world scenario testing

Conclusion

The smart rider safety jacket successfully demonstrates an integrated approach to motorcycle rider protection, combining multiple hazard detection systems with automated emergency response mechanisms. The system provides comprehensive safety coverage through alcohol monitoring, fire detection, and advanced accident recognition, while ensuring immediate physical protection and emergency notification through the airbag system and GPS-enabled alerts.

The STM32 microcontroller effectively coordinates all safety functions, while the 2-cell Li-ion battery ensures reliable portable operation. The project represents a significant advancement in rider safety technology, potentially reducing accident fatalities through faster emergency response and physical impact protection. The integration of multiple safety layers creates a robust protection system that addresses various risks faced by motorcycle riders.

R385 6-12V DC Diaphragm

R385 6-12V DC Diaphragm Based Mini Aquarium Water Pump is an ideal non submersible pump for variety of liquid movement application. It has enough pressure to be used with nozzle to make spray system. The pump can handle heated liquids up to a temperature of 80°C and when suitably powered can suck water through the tube from up to 2m and pump water vertically for up to 3m.

Possible uses/projects include; a small aquarium pump, automatic plant watering system, making a water feature or music activated dancing water features to name but a few. When pumping a liquid the pump runs very quietly. The pump is also capable of pumping air, though when pumping air the pump is quite noisy in comparison.

The R385 requires between 6 – 12V DC and between 0.5 – 0.7A and will deliver its maximum operating values when power is at the upper end of these ranges.

This immersible pump can be used to water your plants, make a fountain or waterfall, and even change your fish tank water. It works quietly with the sound level under 30db. The pump has a filter inside as well as a suction cup which can help stick it to smooth surfaces tightly.

Specifications and Features of R385 6-12V DC Diaphragm Based Mini Aquarium Water Pump:-

Model : R385

Rated Voltage : DC 6V to 12V (1 amps)

Working current: 0.5A to 0.7A (Max)

Power: 4W-7W

Max Lift: 3m

Max Suction: 2m

Max Water Temp: 80 °C

Pump Size: 90mm * 40mm * 35mm approx

Fluid: 0-100 ° C

Input/output tube diameter: outer 8.5mm, inner 6mm approx

Max Current: Up to 2 Amps while starting up

Life: up to 2500 Hours

The maximum flow rate of up to 1 – 3L/min.

No memory effect

Capacity, resistance, Voltage, platform time consistency is good.

Good consistency and low

Battery

This LG INR18650 M26 2600mAh Lithium-Ion Battery gives value for your money. It comes with a rated voltage of 3.7 volts and a capacity of 2600mAh. It is a single cell, compact, and powerful battery cell with 2600 mAh capacity. It is very convenient to install in your project to fulfill the 3.7 Volt requirement with high capacity.

The battery terminals can use in any compatible battery adapter/holder or it can be permanently soldered to your applications power source wires.

Features :

High energy density

High working voltage for single battery cells.

Pollution-free

Long cycle life

self-discharge.

Lightweight, small size

Shape: Cylindrical Battery

Battery Type: Lithium-Ion Battery

High performance and capacity

Flat top to suit many devices fitting.

IR SENSOR

Infrared Obstacle Avoidance IR Sensor Module (Active Low) has a pair of infrared transmitting and receiving tubes. When the transmitted light waves are reflected back, the reflected IR waves will be received by the receiver tube. The onboard comparator circuitry does the processing and the green indicator LED comes to life.

The module features a 3 wire interface with Vcc, GND and an OUTPUT pin on its tail. It works fine with 3.3 to 5V levels. Upon hindrance/reflectance, the output pin gives out a digital signal (a low-level signal). The onboard preset helps to fine-tune the range of operation, the effective distance range is 2cm to 80cm.

Features:

Easy to assemble and use

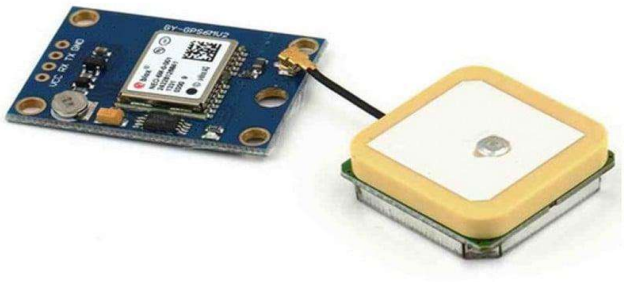
Onboard detection indication

The effective distance range of 2cm to 80cm

A preset knob to fine-tune distance range

If there is an obstacle, the indicator lights on the circuit board.

3.NEO-6M GPS Module with EPROM



If you are wandering to navigate the distance with a drone then this GPS Module is needed for **you. GPS drones are equipped with a GPS module that allows them to know their location relative to a network of orbiting satellite**s. Connecting to signals from these satellites allows the drone to perform functions such as position hold, autonomous flight, return to home, and waypoint navigation. This is a complete GPS module that is based on the **NEO 6M GPS**. This unit uses the latest technology to give the best possible positioning information and includes a larger built-in **25 x 25mm** active GPS antenna with a UART TTL socket. A battery is also included so that you can obtain a GPS lock faster. This is an updated GPS module that can be used with ardupilot mega **v2**. This GPS module gives the best possible position information, allowing for better performance with your Ardupilot or other Multirotor control platform.

The GPS module has serial TTL output, it has four pins: **TX, RX, VCC, and GND**. You can download the u-center software for configuring the GPS and changing the settings and much more. It is really good software

Model	Ublox NEO-6M

Receiver Type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS
Input Supply Voltage (VDC)	2.7 ~ 6
Main Chip	NEO-6
Sensitivity (dBm)	-160 156 Cold Start (without aiding): -147 dBm Tracking & Navigation: -161 dBm
Navigation Update Rate	5Hz
Position Accuracy (Meter)	2
Operating Temperature Range (°C)	-24 to 84
Tracking Sensitivity (dBm)	-161 dBm
Avg Cold Start Time (s)	27
Warm Start Time (s)	27
Maximum Speed	500 M/s
Dimensions (mm) LxWxH	Antenna – 25 x 25 x 7 GPS Board – 22 x 30 x 4
Weight (gm)	12
Cable Length (cm)	10
Shipment Weight	0.016 kg
Shipment Dimensions	8 × 6 × 4 cm

SIM800L GSM Module

This is Small SIM800L GPRS GSM Module Micro SIM Card Core Board Quad-band TTL Serial Port with the antenna, in this module two antennas have been included.

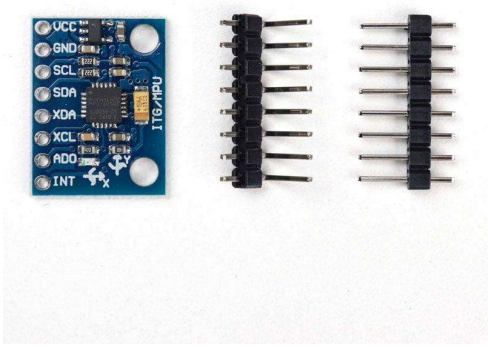
SIM800L GSM/GPRS module is a miniature GSM modem, which can be integrated into a great number of IoT projects. You can use this module to accomplish almost anything a normal cell phone can; SMS text messages, make or receive phone calls, connecting to the internet through GPRS, TCP/IP, and more! To top it off, the module supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world

First is made of wire (which solders directly to NET pin on PCB) – very useful in narrow places. Second – PCB antenna – with double-sided tape and attached pigtail cable with IPX connector. This one has better performance and allows to put your module inside a metal case – as long the antenna is outside.

Features :

1. TTL serial port for serial port, you can link directly to the micro-controller.
2. Don’t need MAX232.
3. Power module automatically boots, homing network.
4. Onboard signal lights all the way.
5. It flashes slowly when there is a signal, it flashes quickly when there is no signal.

MPU-6050 3-Axis Accelerometer and Gyro Sensor



The **MPU-6050** 3-Axis Accelerometer and Gyro Sensor module use **MPU-6050** which is a little piece of motion processing tech. The MPU6050 datasheet is available in the attachment section to get this visit.

There are different types of magnetometer are available the basic differences between **MPU6000**, **MPU6050** and **MPU6500** is:

MPU6000	**MPU6050**	**MPU6500**
1. A sampling rate of 8 kHz.	1. A sampling rate of 8 kHz.	1. A sampling rate of 32 kHz.
2. Supports SPI and I^2C interface. But I^2C is too slow to handle 8 kHz gyro update.	2. Supports I^2C interface communication protocol.	2. Supports both I^2C and SPI interface.
3. Vibration Sensitivity is better than MPU6500.	3. Vibration Sensitivity is better than MPU6500 but the speed of operation is less than MPU6000.	3. It is more susceptible to vibrations, so the need for some vibration isolation methods. It is faster than both MPU.

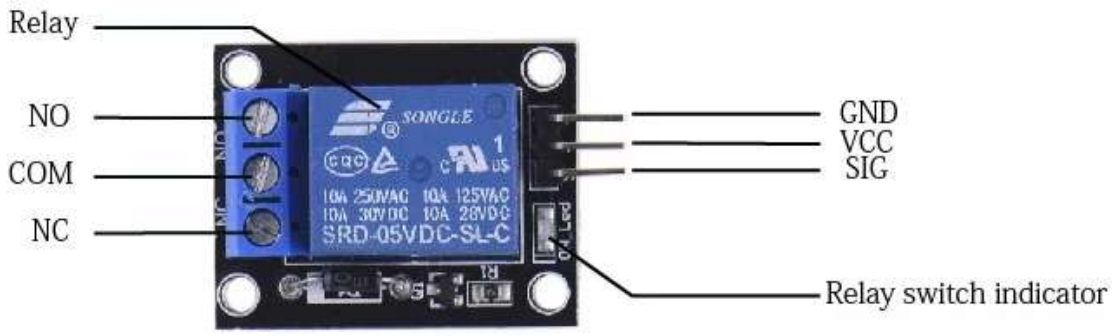
The **MPU6050** devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon together with an onboard Digital Motion Processor (DMP) capable of processing

complex 9-axis MotionFusion algorithms. To get more insights about specifications visit the MPU6050 datasheet.

Features :

1. Chip built-in 16bit AD converter, 16-bit data output.
2. I2C Digital-output of 6 or 9-axis MotionFusion data in rotation matrix, quaternion, Euler Angle, or raw data format.
3. Selectable Solder Jumpers on CLK, FSYNC, and AD0.
4. Digital Motion Processing™ (DMP™) engine offloads complex MotionFusion, sensor timing synchronization, and gesture detection.
5. Embedded algorithms for run-time bias and compass calibration. No user intervention is required.
6. Digital-output temperature sensor.

5V 1 Channel Relay Module



This 1-channel 5V control Single-Pole Double-Throw (SPDT) High-level trigger AC power relay board can be controlled directly via a microcontroller and switch up to 10A at 250 VAC. The inputs of 1 Channel 5V Relay Module are isolated to protect any delicate control circuitry.

The default state of the relay when the power is off for COM (Power) to be connected to NC (Normally Closed). This is the equivalent of setting the relay board IN pin to HIGH (has +5V sent to it).

No. of Channels	1
Trigger Voltage (VDC)	5
Trigger Current (mA)	20
Switching Voltage (VAC)	250@10A
Switching Voltage (VDC)	30@10A
Length (mm)	38
Width (mm)	26
Height (mm)	18
Weight (gm)	13
Shipment Weight	0.017 kg
Shipment Dimensions	7 × 5 × 2 cm

,

MQ-3 Alcohol Detector Gas Sensor Module

This MQ-3 Alcohol Detector Gas Sensor Module is simple to use. MQ-3 Alcohol Detector Module which can sense the presence of Alcohol in the air. The module uses our MQ-3 sensor. It simplifies the interface to the odd pin spacing of the sensor and provides an interface through standard 4 x 0.1″ header pins.

It provides an analog output corresponding to the concentration of the gas in the air and an easy to use digital output.

The onboard potentiometer can be used to set the maximum gas concentration beyond which the digital output gets triggered. Just power the module with 5V set the threshold and you can start getting the gas concentration of the air around the sensor! An onboard LED signals the presence of any gas. Perfect sensing alcohol concentration on your breath just likes a common breathalyzer.

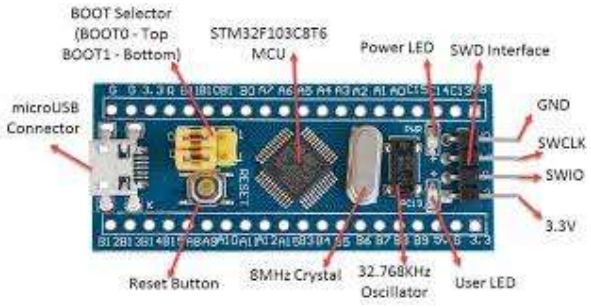
The digital output can be easily interfaced to microcontrollers and other circuits. The analog output can be hooked up to an ADC of a microcontroller to get a wide range of sensor readings.

Features :

1. Signal output light indicator.
2. Analog and level signal output.
3. TTL output valid signal is low. (When the output low signal light can be directly connected microcontroller or relay module).

4. The carbon monoxide detection with better sensitivity.
5. It has a long service life and reliable stability.
6. Rapid response recovery features.2

STM32



This is STM32F103C8T6 Minimum System Board STM32 ARM Core Module. This board is a low-cost Minimum System Development Board for ARM Microcontroller STM32F103C8T6.

Board is suitable for learners that want to learn STM32 microcontroller with ARM Cortex-M3 32-bit core.

Specifications and Features:-

- 100% Brand new and high-quality STM32 Board
- 72MHz work frequency.
- 64K flash memory, 20K SRAM.
- 2.0-3.6V power, I/O.
- Reset(POR/PDR).
- 4-16MHz crystal.
- Onboard Mini USB interface, you can give the board power supply and USB communication.

SOFTWARE DESCRIPTION

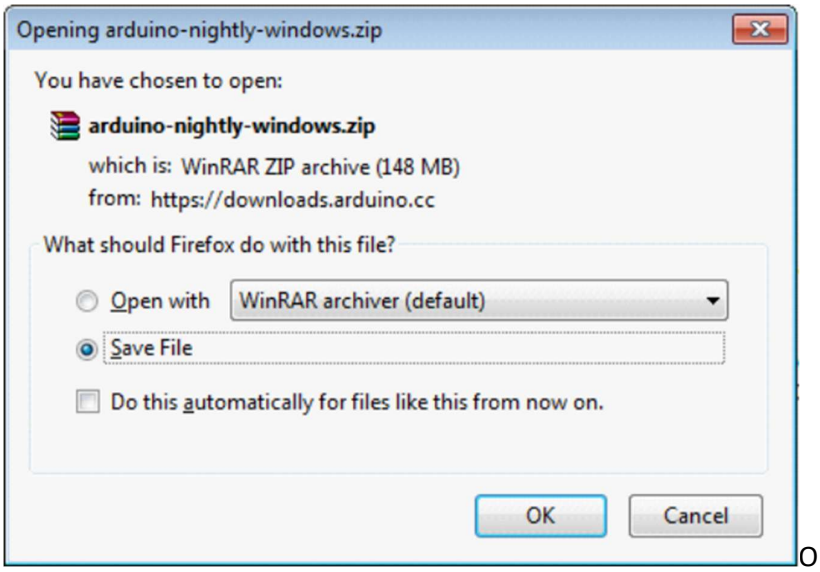
Arduino ide is the software used to write-compile-upload program to arduino.Its a open source software

- **Procedure to Install Arduino Software (IDE)**

Step 1 — First we must have an Arduino board and a USB cable. In case we use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, we will need a standard USB cable (A plug to B plug), the kind we would connect to a USB printer as shown in Fig. 3.1.5.1 (a). In case we use Arduino Nano, we will need an A to Mini-B cable instead as shown in Fig. 3.1.5.1 (b).

Fig 3.1.5.1 (b) A to Mini-B Cable

Step 2 — Download Arduino IDE Software.



One can get different versions of Arduino IDE from the Download page on the Arduino Official website. We must select our software, which is compatible with our operating system (Windows, IOS, or Linux). Unzip the file, after downloading it completely.

Fig. 3.1.5.2 Downloading Arduino IDE

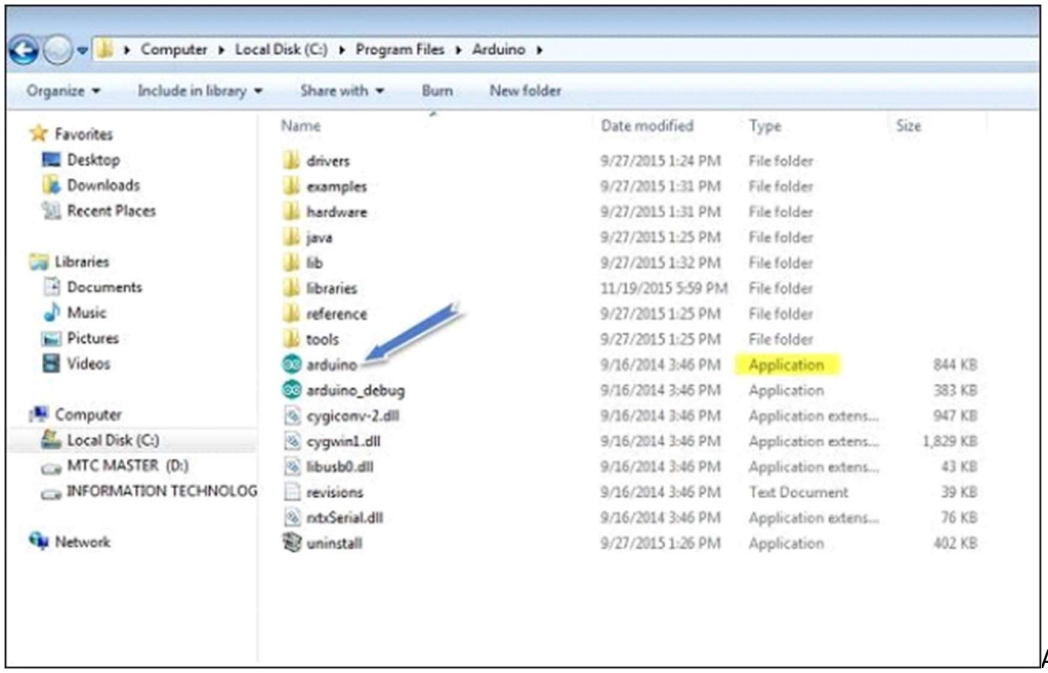
Step 3 — Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of

plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 — Launch Arduino IDE.



After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double- click the icon to start the IDE.

Fig. 3.1.5.4 Launching Arduino IDE

Step 5 — Open your first project.

Once the software starts, we have two options:

- Create a new project.

  To create a new project, select File →**New,** as shown in Fig. 3.1.5.5 (a).

- Open an existing project example.

  To open an existing project example, select File → Example → Basics → Blink, as shown in Fig. 3.1.5.5 (b).

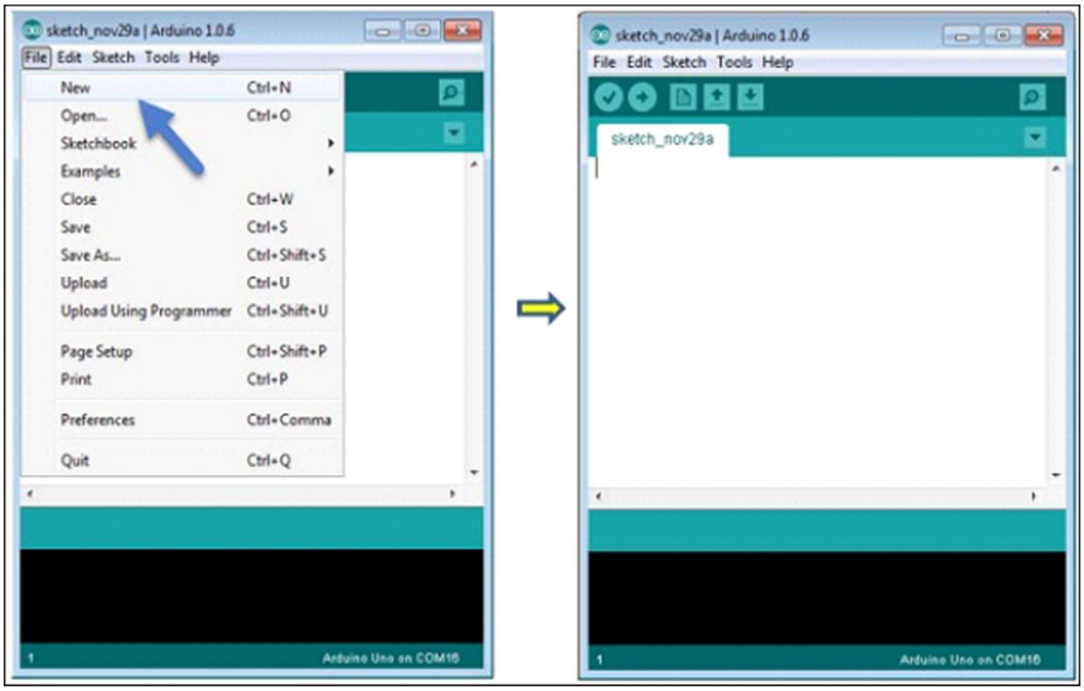


Fig. 3.1.5.5 (a) Creating a New Project

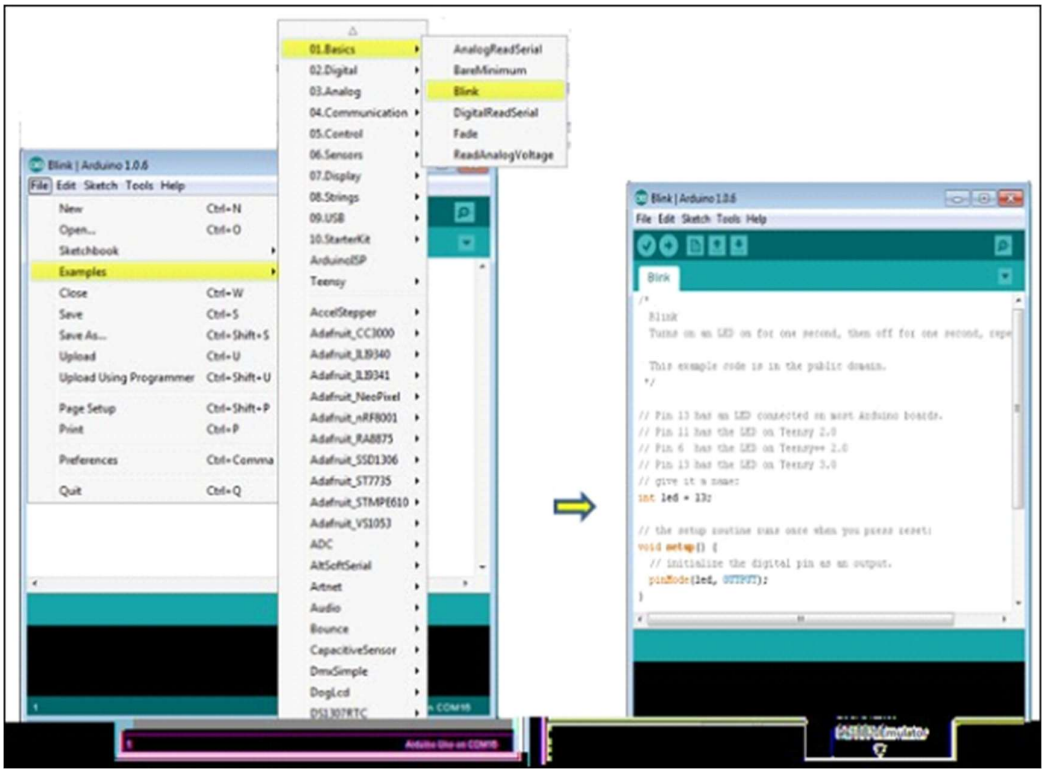


Fig. 3.1.5.5 (b) Opening an Existing Project

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. We can select any example from the list.

Step 6 — Select the respective stm32f1 board.

To avoid any error while uploading our program to the board, we must select the correct Arduino board name, which matches with the board connected to our computer.

Go to Tools → Board and select the board.

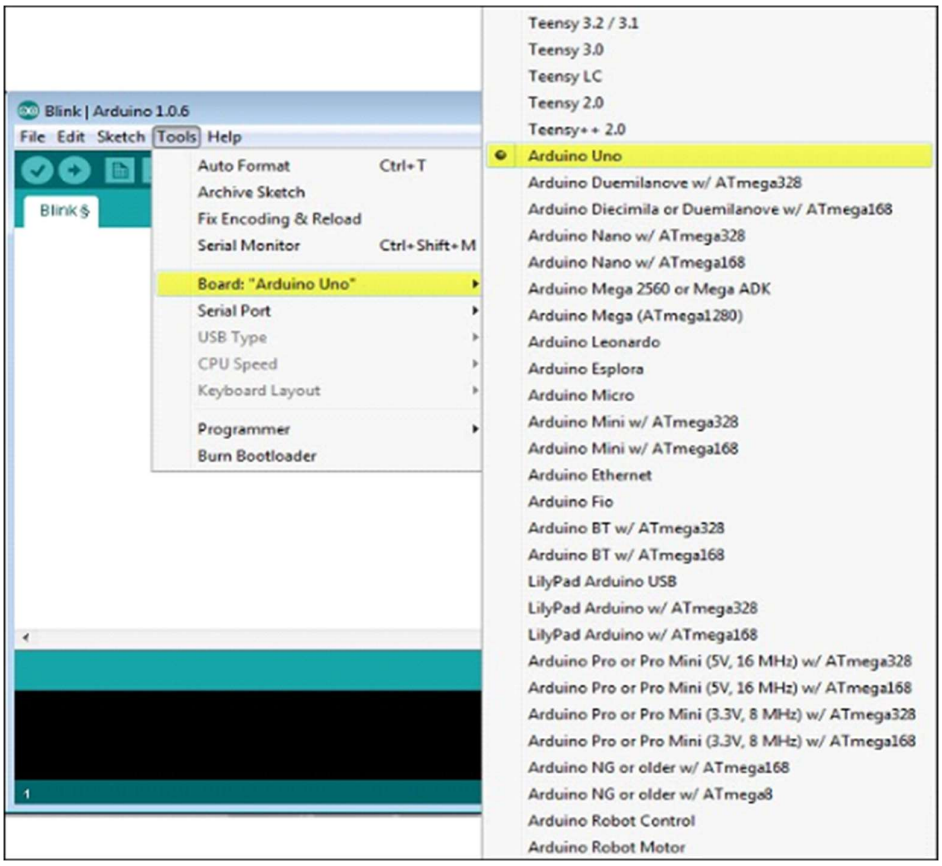


Fig. 3.1.5.6 Selecting the Arduino Board

Step 7 — Select the serial port.

Select the serial device of the Arduino board. Go to **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 Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

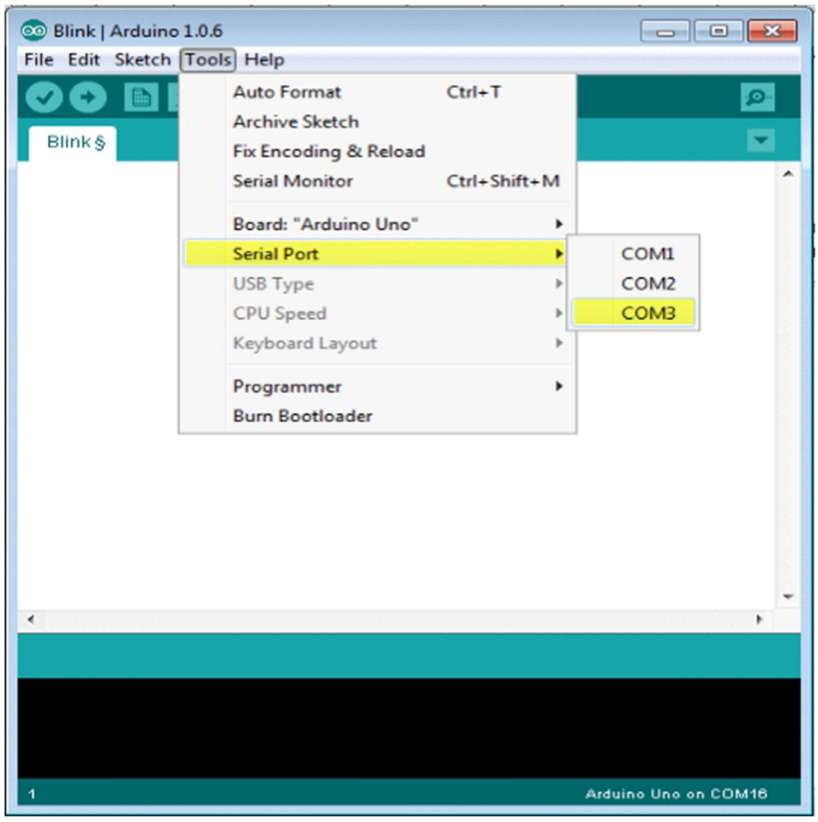
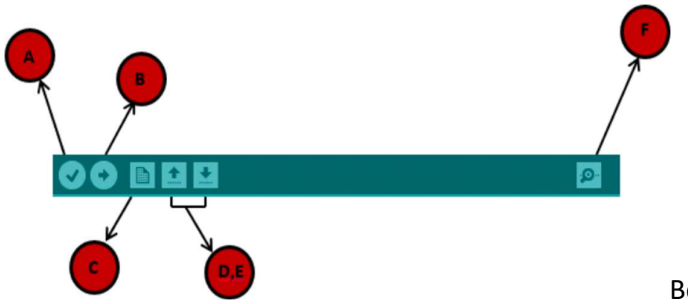


Fig. 3.1.4.7 Selecting the Serial Port

Step 8 — Upload the program to the board.



Before explaining how to upload our program to the board, we should know the function of each symbol appearing in the Arduino IDE toolbar.

Fig. 3.1.4.8 Arduino IDE Toolbar

A — Used to check if there is any compilation error. **B** — Used to upload a program to the Arduino board. **C** — Shortcut used to create a new sketch.

D — Used to directly open one of the example sketch.

E — Used to save your sketch.

F — Serial monitor used to send and receive the serial data from the board.

Now, simply click the "Upload" button in the environment. Wait few seconds; we will 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.

- **Program Structure**

In this section, we will study in depth, the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL. Arduino programs are called sketch.

Arduino programs can be divided in three main parts: **Structure, Values** (variables and constants), and **Functions**. Let us learn about the Arduino software program, step by step, and how to write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consist of two main functions:

- Setup( ) function
- Loop( ) function

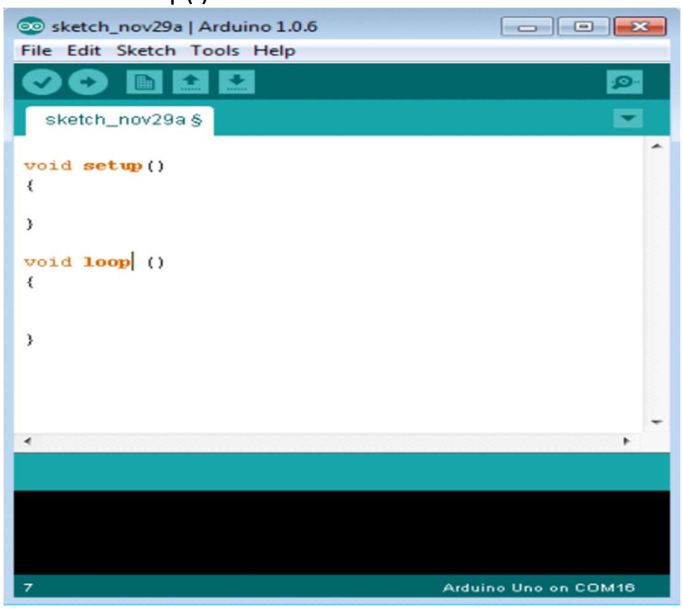


Fig. 3.1.6 Structure of a Sketch

The **setup ()** function is called when a sketch starts. It is used to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

After creating a **setup ()** function, which initializes and sets the initial values, the **loop ()** function does precisely what its name suggests, and loops consecutively, allowing our program to change and respond. It actively controls the Arduino board.

- **Programming using Embedded C**

C is a high-level programming language intended for system programming. Embedded C is an extension that provides support for developing efficient programs for embedded devices. Yet, it is not a part of the C language. In our Internship program, we employed Embedded C programs to write sketches to be dumped on Arduino Uno.

- **Introduction to Embedded C**

Embedded C programming language is an extension to the traditional C programming language that is used in embedded systems. The embedded C programming language uses the same syntax and semantics as the C programming language.

The only extension in the Embedded C language from normal C Programming Language is the I/O Hardware Addressing, fixed-point arithmetic operations, accessing address spaces, etc.

- **Basic Structure of Embedded C Program:**

The embedded C program has a structure similar to C programming. The five layers of Embedded C programming structure are:

- Comments
- Pre-processor directives
- Global declaration
- Local declaration
- Main function()

The whole code follows the below outline. This is the basic structure of the embedded c program. Each code has a similar outline. Now let us learn about each of this layer in detail.

Outline of an Embedded C code is as shown below:

- Multiline Comments Denoted using /*……*/
- Single Line Comments Denoted using //
- Pre-processor Directives #include<…> or #define
- Global Variables Accessible anywhere in the program
- Function Declarations Declaring Function
- Main Function Main Function, execution begins here

7. {

- Local Variables Variables confined to main function
- Function Calls Calling other Functions
- Infinite Loop Like while (1) or for (;;)

- Statements . . . . .

12. ….

13. ….

14. }

15. Function Definitions Defining the Functions

16. {

- Local Variables Local Variables confined to this Function
- Statements . . . . .

19. ….

20. ….

21. }

- **Comment Section:** Comments are simple readable text, written in code to make it more understandable to the reader. Usually comments are written in // or /* */.

  Example: //Test program

- **Pre-processor Directives Section:** The Pre-Processor directives tell the compiler which files to look in to find the symbols that are not present in the program.

For Example, in 8051 Keil compiler we use,

- #include<reg51.h>

- **Global Declaration Section:** The global variables and the user-defined functions are declared in this part of the code. They can be accessed from anywhere.

  1. void delay (int);

- **Local Declaration Section:** These variables are declared in the respective functions and cannot be used outside the main function.
- **Main Function Section:** Every C programs need to have the main function. So does an embedded C program. Each main function contains 2 parts. A declaration part and an Execution part. The declaration part is the part where all the variables are declared. The execution part begins with the curly brackets and ends with the curly close bracket. Both the declaration and execution part are inside the curly braces.

  Example:

  1. void main(void) // Main Function 2. {

```
3.      P1 = 0x00;
4.      while(1)
5.      {
6.      P1 = 0xFF;
7.      delay(1000);
8.      P1 = 0x00;
9.      delay(1000);
10.     }
11.     }
```

- **Function Definition Section:** The function is defined in this section.

- **Arduino Code libraries**
- **Library Structure**

- A library is a folder comprised with C++ (.cpp) code files and C++ (.h) header files.
- The .h file describes the structure of the library and declares all its variables and functions.
- The .cpp file holds the function implementation.

- **Importing Libraries**

The first thing to do is to find the library we want to use out of the many libraries available online. After downloading it to our computer, we just need to open Arduino IDE and click on Sketch > Include Library > Manage Libraries. We can then select the library we want to import. Once the process is complete the library will be available in the sketch menu.

- **Arduino Code Explanation**

Arduino code is written in C++ with an addition of special methods and functions. C++ is a human-readable programming language. When we create a sketch (the name given to Arduino code files), it is processed and compiled to machine language.

- **Code Structure**

The basic concepts which one should know to write a program on Arduino IDE are discussed below:

- **Libraries**

In Arduino, much like other leading programming platforms, there are built-in libraries that provide basic functionality. In addition, its possible to import other libraries and expand the Arduino board capabilities and features. These libraries are roughly divided into libraries that interact with a specific component or those that implement new functions.

To import a new library, we need to go to Sketch > Import Library

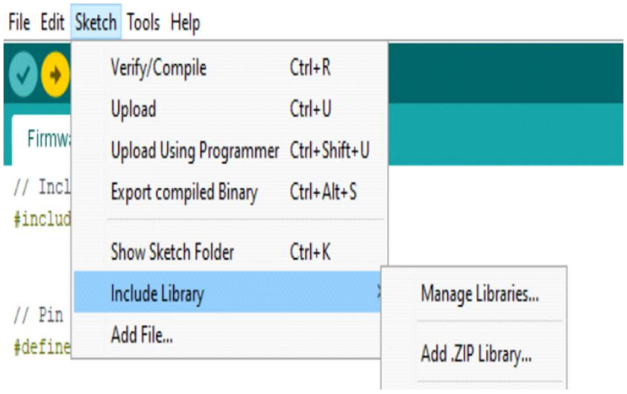


Fig. 3.2.2.1 Including Libraries on IDE

In addition, at the top of our file, we have to use #include to include external libraries. We can also create custom libraries to use in isolated sketches.

- **Pin Definitions**

To use the Arduino pins, we need to define which pin is being used and its functionality. A convenient way to define the used pins is by using:

#define pinName pinNumber.

The functionality is either input or output and is defined using the pinMode () method in the setup section.

- **Declarations**

- **Variables**

Whenever were using Arduino, we need to declare global variables and instances to be used later on. In a nutshell, a variable allows us to name and store a value to be used in the future. For example, we would store data acquired from a sensor in order to use it later. To declare a variable we simply define its type, name and initial value. Its worth mentioning that declaring global variables isnt an absolute necessity. However, its advisable to declare variables to make it easy to utilize our values further down the line.

- **Instances**

In software programming, a class is a collection of functions and variables that are kept together in one place. Each class has a special function known as a constructor, which is used to create an instance of the class. In order to use the functions of the class, we need to declare an instance for it.

- **Setup()**

Every Arduino sketch must have a setup function. This function defines the initial state of the Arduino upon boot and runs only once.

Here well define the following:

- Pin functionality using the pinMode function
- Initial state of pins
- Initialize classes
- Initialize variables
- Code logic

- **Loop()**

The loop function is also a must for every Arduino sketch and executes once setup () is complete. It is the main function and as its name hints, it runs in a loop over and over again. The loop describes the main logic of our circuit.

- **Code Logic**

The basic Arduino code logic is an if-then structure and can be divided into 4 blocks:

- **Setup** - will usually be written in the setup section of the Arduino code, and performs things that need to be done only once, such as sensor calibration.

- **Input** - at the beginning of the loop, read the inputs. These values will be used as conditions (if) such as the ambient light reading from an LDR using analogRead ().
- **Manipulate Data** - this section is used to transform the data into a more convenient form or perform calculations. For instance, the AnalogRead () gives a reading of 0-1023 which can be mapped to a range of 0-255 to be used for PWM. (See analogWrite ())
- **Output** - this section defines the final outcome of the logic (then) according to the data calculated in the previous step. Looking at an example of the LDR and PWM, turns on an LED only when the ambient light level goes below a certain threshold.

- **From Software to Hardware**

There is a lot to be said of Arduinos software capabilities, but its important to remember that the platform is comprised of both software and hardware. The two work in tandem to run a complex operating system.

Code → Compile → Upload → Run

At the core of Arduino, is the ability to compile and run the code.

After writing the code in the IDE we need to upload it to the Arduino. Clicking the Upload button (the right-facing arrow icon), will compile the code and upload it if it passed compilation. Once the upload is complete, the program will start running automatically.