ACKNOWLEDGEMENT

We, take this opportunity to express our sincere gratitude to Prof. Murari Shaw, Department of Electronics and Communication Engineering, for this guidance and valuable suggestions during the preparation of the project.

We would like to acknowledge the help rendered by our HOD Prof. (Dr.) Malay Ganguly and all other faculty members of the Department of Electronics & Communication Engineering.

We are extremely thankful to Prof. (Dr.) Amlan Kusum Nayak (Principal, Institute of Engineering & Management) and the college authorities for their whole hearted support.

We would also like to extend our sincere appreciation to the non-teaching staff as well for their cooperation.

CONTENTS

ABSTRACT

This synopsis presents a design and prototype implementation of new home automation system that uses WiFi technology as a network infrastructure connecting its parts. The proposed system consists of two main components; the first part is the server (web server), which presents system core that manages, controls, and monitors users' home. Users and system administrator can locally (LAN) or remotely (internet) manage and control system code. Second part is hardware interface module, which provides appropriate interface to sensors and actuator of home automation system. Unlike most of available home automation system in the market the proposed system is scalable that one server can manage many hardware interface modules as long as it exists on WiFi network coverage. System supports a wide range of home automation devices like power management components, and security components. The proposed system is better from the scalability and flexibility point of view than the commercially available home automation systems.

OBJECTIVE

The main objective of this project is to design and implement a cheap and open source home automation system that is capable of controlling and automating most of the house appliances.This application is an easy and manageable web interface for user to run Home Automation System.

In this project we have integrated technologies like Android with Wi-Fi to execute Home Automation System. We designed user Interfaces using Android because Android operating systems are capturing most of the mobile market. It has technical advantages of scalability, flexibility, availability, security and its ease of use for users.

The aim to take Android as platform is becausepeople are familiar as many applications are launched in Android. Android provides interactive graphical user interface which makes an application easy to use for users.

In this application, we used fans, bulbs etc. depicted graphically for better understanding of the users. Users can switch ON/OFF any appliances like fan, tube lights etc. as per their convenience through mobile application. They can also check the status of appliances even when they are not at home. This application is scalable to add or delete appliances as per user's requirement.

In this application we embedded features like gas leakage alerts to user by sending simple text message on user's phone. If there is a gas leakage, our system will sense it and send the signals immediately to server. Server will send message to user mobile application connected with server through Wi-Fi. User can take immediate action on receiving SMS from server by automatically turning off the cylinder valve and opening windows through Android application as per user's command. In addition, feature like fire alarm is embedded to sense increase of temperature above threshold value. Server will take appropriate action by sending message via Android application to user. To keep home safe from burglary this project behaves in same fashion.

We have selected Wi-Fi technology to be used in this project because it will keep Home Automation System active and user can interact with server even if user is not present at home.

Wi-Fi is available with multiple ranges, ranging from 150 feet (46m) for indoor and 300 feet (92m) for outdoor. The system maintains log information as well regarding the units consumed by various appliances which can help the user acquire knowledge about individual device power consumption.

OVERVIEW

Home automation or domotics is **building automation** for a home, called a smart home or smart house. It involves the control and automation of lighting, heating (such as **smart thermostats**), ventilation, air conditioning (**HVAC**), and security, as well as **home appliances** such as washer/dryers, ovens or refrigerators/freezers. **Wi-Fi** is often used for remote monitoring and control. Home devices, when remotely monitored and controlled via the Internet, are an important constituent of the **Internet of Things**. Modern systems generally consist of switches and sensors connected to a central hub sometimes called a "gateway" from which the system is controlled with a **user interface** that is interacted either with a wall-mounted terminal, mobile phone software, **tablet computer** or a web interface, often but not always via Internet cloud services.

While there are many competing vendors, there are very few worldwide accepted industry standards and the smart home space is heavily fragmented. Popular **communications protocol** for products include **X10, Ethernet, RS-485, 6LoWPAN, Bluetooth LE (BLE), ZigBee** and **Z-Wave**, or other proprietary protocols all of which are incompatible with each other. Manufacturers often prevent independent implementations by withholding documentation and by litigation.

The home automation market was worth US$5.77 billion in 2013, predicted to reach a market value of US$12.81 billion by the year 2020.

INTRODUCTION

Nowadays, home and building automation systems are used more and more. On the one hand, they provide increased comfort especially when employed in a private home. On the other hand, automation systems installed in commercial buildings do not only increase comfort, but also allow centralized control of heating, ventilation, air condition and lighting. Hence, they contribute to an overall cost reduction and also to energy saving which is certainly a main issue today. Existing, well-established systems are based on wired communication. Examples include BAC-net, Lon-Works and KNX. Employing a traditional wired automation system does not pose a problem as long as the system is planned before and installed during the physical construction of the building. If, however, already existing buildings should be augmented with automation systems, this requires much effort and mush cost since cabling is necessary. Obviously, wireless systems can come to help here. In the past few years, wireless technologies reached their breakthrough. Wireless based systems used every day and everywhere, range from wireless home networks and mobile phones to garage door openers. As of today, little comparative research of wireless automation standards has been done, although such knowledge would provide valuable information to everyone looking for the most suitable system for given requirements.

FEATURES & BENEFITS

In recent years, wireless systems like WLAN have become more and more common in home networking. Also in home and building automation systems, the use of wireless technologies gives several advantages that could not be achieved using a wired network only.

1) Reduced installation costs: First and foremost, installation costs are significantly reduced since no cabling is necessary. Wired solutions require cabling, where material as well as the professional laying of cables (e.g. into walls) is expensive.

2) Easy deployment, installation, and coverage: Wireless nodes can be mounted almost anywhere. In adjacent or remote places, where cabling may not be feasible at all, e.g., a garden house or the patio, connection to the home network is accomplished instantly by simply mounting nodes in the area. Hence, wireless technology also helps to enlarge the covered area.

3) System scalability and easy extension: Deploying a wireless network is especially advantageous when, due to new or changed requirements, extension of the network is necessary. In contrast to wired installations, additional nodes do not require additional cabling which makes extension rather trivial. This makes wireless installations a seminal investment.

4) Aesthetical benefits: As mentioned before, placement of wireless nodes is easy. Apart from covering a larger area, this attribute helps to full aesthetical requirements as well. Examples include representative buildings with all-glass architecture and historical buildings where design or conservatory reasons do not allow laying of cables.

5) Integration of mobile devices: With wireless networks, associating mobile devices such as PDAs and Smartphones with the automation system becomes possible everywhere and at any time, as a device's exact physical location is no longer crucial for a connection (as long as the device is in reach of the network).

Typical examples include an engineer who connects to the network, performs a particular management task, and disconnects after having finished the task; or control of blinds using a remote control.For all these reasons, wireless technology is not only an attractive choice in renovation and refurbishment, but also for new installations.

PROPOSED SYSTEM

A. FEATURES

The Entire project consists of two main phases i.e. Hardware and Software. User has the central control over home appliances by using Android phone application. User commands through Android application whose signal is given to PC via Wi-Fi. PC has the sever program deployed on it. Server is configured to handle both hardware and software modules. Microcontroller using serial communication port interacts with server.

As per user's command particular appliance is operated (ON/OFF & precise control). Server keeps record of log information which is provided to user on demand and temperature readings regularly updated on user's application. In case if gas leak or fire hazard occurs, it will send notifications to user about it, so necessary actions could be taken and hazards can be avoided. Through Wi-Fi, server and user application is connected. Wi-Fi is chosen to improve system security (by using secure Wi-Fi connection), and to increase system mobility and scalability. In case when no one is present in the room, the appliances will automatically get switch OFF, thus saving electricity.

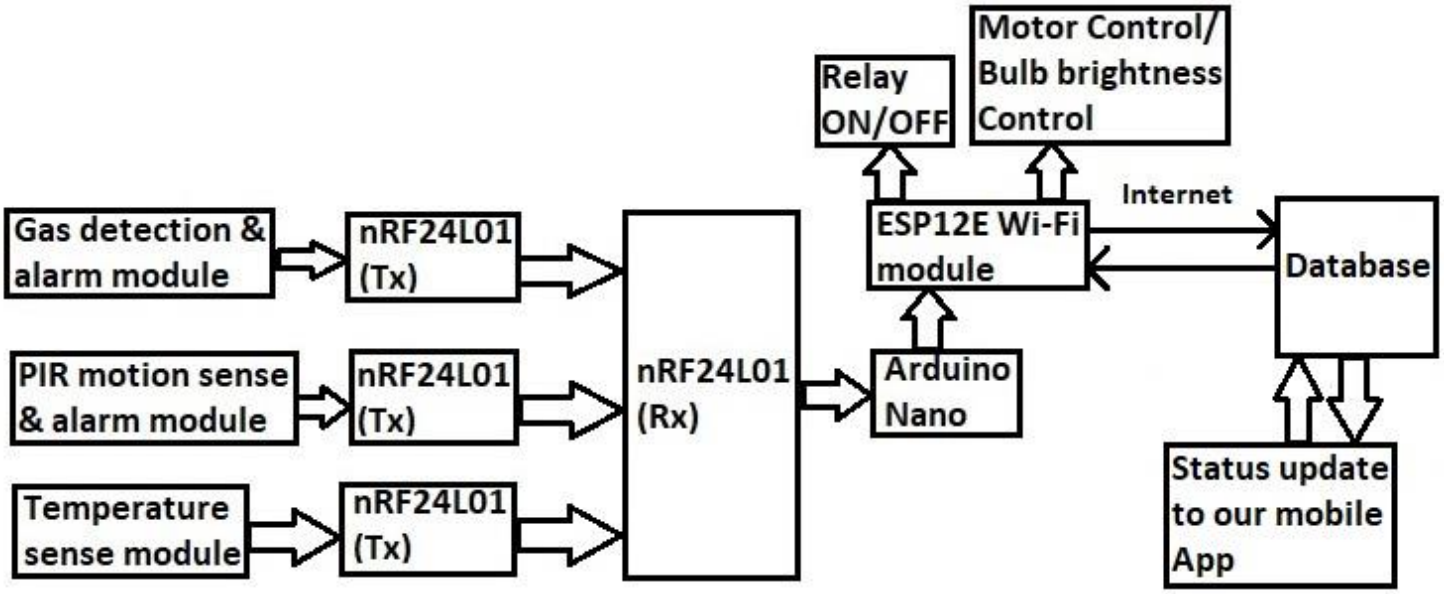


Figure1: Block Diagram

B. SYSTEM DESIGN

- ***Hardware Design***

The hardware circuit consists of ESP12E, a low-cost **Wi-Fi** chip with full TCP/IP stack and **MCU (microcontroller unit)** capability, Arduino pro MINI & NANO, nRF24L01 2.4GHz RF communication link, temperature sensor (DHT22), LPG sensor (MQ6), Relays and PIR motion sensors.

ESP12E which consists of ESP8266,wifi chip has L106 32-bit **RISC** microprocessor core based on the **Tensilica** Xtensa Diamond Standard 106Micro running at 80 MHz, 64 KiB of instruction RAM, 96 KiB of data RAM , 16 MiB External QSPI flash is supported (512 KiB to 4 MiB typically included). It has **IEEE 802.11** b/g/n **Wi-Fi, Integrated TR switch, balun, LNA, power amplifier** and **matching network**. It provides **WEP** or **WPA/WPA2** authentication, or open networks. It includes 16 **GPIO** pins, **SPI, I^2C** (software implementation), **I^2S** interfaces with DMA (sharing pins with GPIO), **UART** on dedicated pins, plus a transmit-only UART can be enabled on GPIO2,10-bit **ADC (successive approximation ADC)**.

On AC power supply of 230V, the Bridge rectifier (D1) converts it into DC. To get a constant output voltage of 12V DC, power adapter is used and 7805 is used since relay circuitry requires 5V supply whereas 3.3V regulator is used for wifi chip and nRF24L01 working. Capacitors, (electrolytic or ceramic) and resistors with their specific values are mounted as per the requirements. LED's mounted on the circuit indicates whether the circuit is working properly.

The whole project is divided into 4 parts/modules, whereas 3 modules are done in the same board to reduce the cost of the project and other module is made separately. So, the 3 modules are-

- Gas sensing and alarm system
- Motion sensing and alarm system
- temperature sensing system

The other module doesn't have any name , but it is main among 4 modules, so, we call it 'Main module'.

Detailed description of these modules are given below.

Basic components used in this project :

1. Arduino Pro MINI

The original, true-blue Arduino is open-source hardware, which means anyone is free to download the design files and spin their own version of the popular development board. SparkFun has jumped on this opportunity and created all sorts of Arduino variants, each with their own unique features, dimensions, and applications. Now one of those variants has landed in your hands; congratulations! It's a wild world out there in microcontroller-land, and you're about to take your first step away from the wonderful – though sometimes stifling – simplicity of the Arudino Pro Mini.

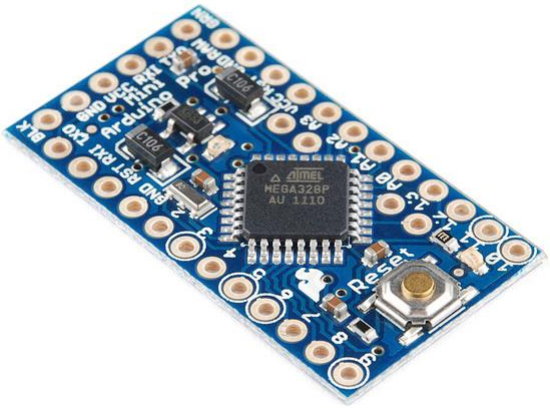


fig. Arduino pro mini

Tech Specs

Microcontroller	ATmega328p
Board Power Supply	3.35 -12 V (3.3V model) or 5 - 12 V (5V model)
Circuit Operating Voltage	3.3V or 5V (depending on model)
Digital I/O Pins	14
PWM Pins	6
UART	1
SPI	1
I2C	1
Analog Input Pins	6
External Interrupts	2
DC Current per I/O Pin	40 mA
Flash Memory	32KB of which 2 KB used by bootloader *
SRAM	2 KB *
EEPROM	1 KB *
Clock Speed	8 MHz (3.3V versions) or 16 MHz (5V versions)

The FTDI Basic will be used to program (and power) the Pro Mini. The headers are optional, but they're our preferred way to interface other devices to the Pro Mini.

Assembly of the Pro Mini also requires soldering. This is a great place to start soldering, if you've never done it before! The joints are all easy, through-hole jobs.

What It Is (and Isn't)

So what differentiates the Arduino Pro Mini from the Arduino Uno? Well, the most obvious difference is the form factor. The Pro Mini's pretty…mini, measuring in at just 1.3x0.70". It's about ⅛th the size of the Arduino Uno. The compact size is great for projects where you may need to fit the Arduino into a tiny enclosure, but it also means that the Pro Mini is not physically compatible with Arduino shields (you could still hard-wire the Mini up to any Arduino shield).

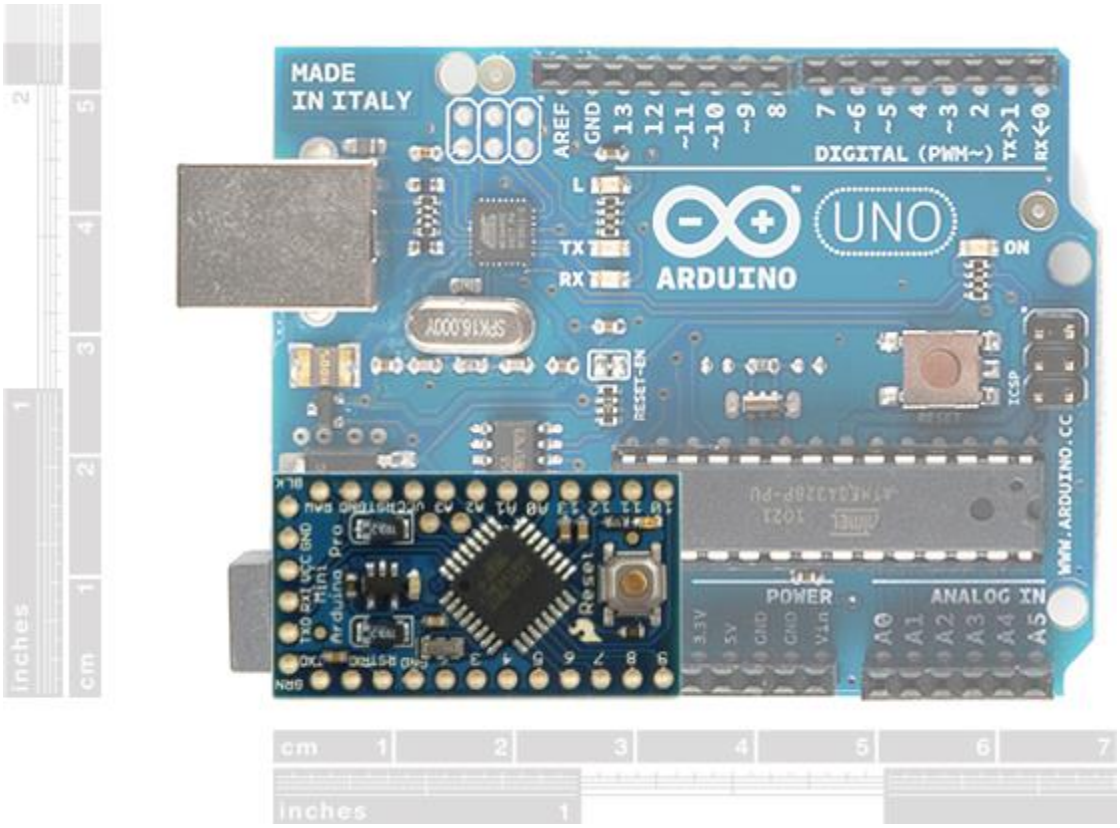


fig. Comparing the size of a standard Arduino Uno with the (aptly named) Pro Mini.

The Mini packs almost as much microprocessor-punch as the regular Arduino, but there are a few major hardware changes you should be aware of before you start adapting your project to the Mini. The first glaring hardware difference is the voltage that the Mini operates at: 5v/3.3V. Unlike the Arduino Uno, which has both a 5V and 3.3V regulator on board, the Mini only has one regulator. This means that if you've got peripherals that only work at 5V, you might have to do some level shifting before you hook it up to the Pro Mini (or you could go for the 5V variant of the Pro Mini).

Another major variation from the standard Arduino lies in the speed at which the ATmega328 runs. The Pro Mini 3.3V runs at 8MHz/ 5Vruns at 16MHz. We put a slower resonator on the Mini to guarantee safe operation of the ATmega. That said, don't let the slower speed scare you away from

using the Mini; 8MHz is still plenty fast, and the Mini will still be capable of controlling almost any project the Arduino Uno can.

Speed Grades

Maximum frequency is dependent on V_{CC}. As shown in Figure 28-1, the Maximum Frequency vs. V_{CC} curve is linear between $1.8V < V_{CC} < 2.7V$ and between $2.7V < V_{CC} < 4.5V$.

Figure 28-1. Maximum Frequency vs. V_{CC}

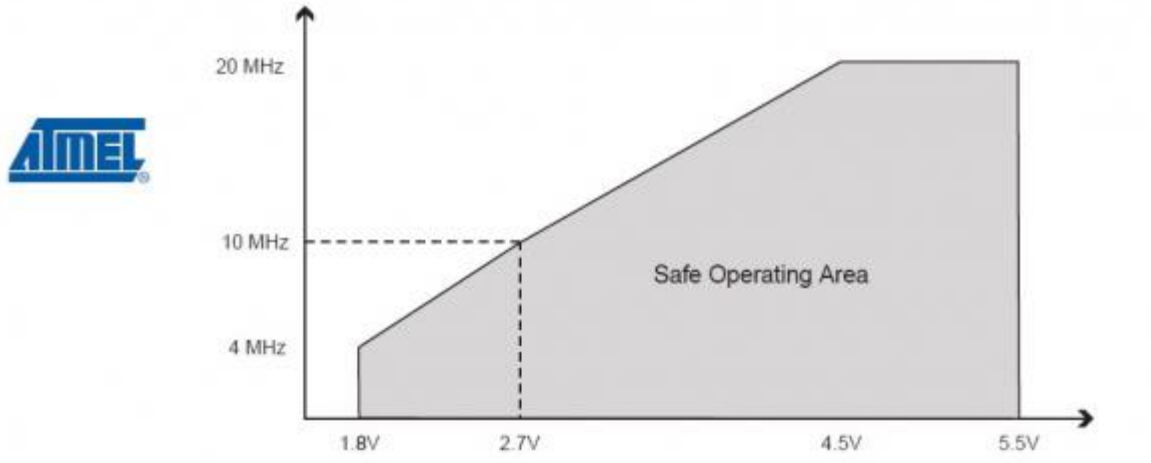


fig. speed grades

One last missing piece of hardware is the Atmega16U2-based USB-to-Serial converter, and the USB connector that goes with it. All of the USB circuitry had to be eliminated for us to make the Pro Mini as small as possible. The absence of this circuit means an external component, the FTDI Basic Breakout, is required to upload code to the Arduino Pro Mini.

fig. Serial connector

Schematic and Pin-out

The schematic of the Pro Mini can be broken down into three blocks: the voltage regulator, the ATmega328 and supporting circuitry, and the headers.

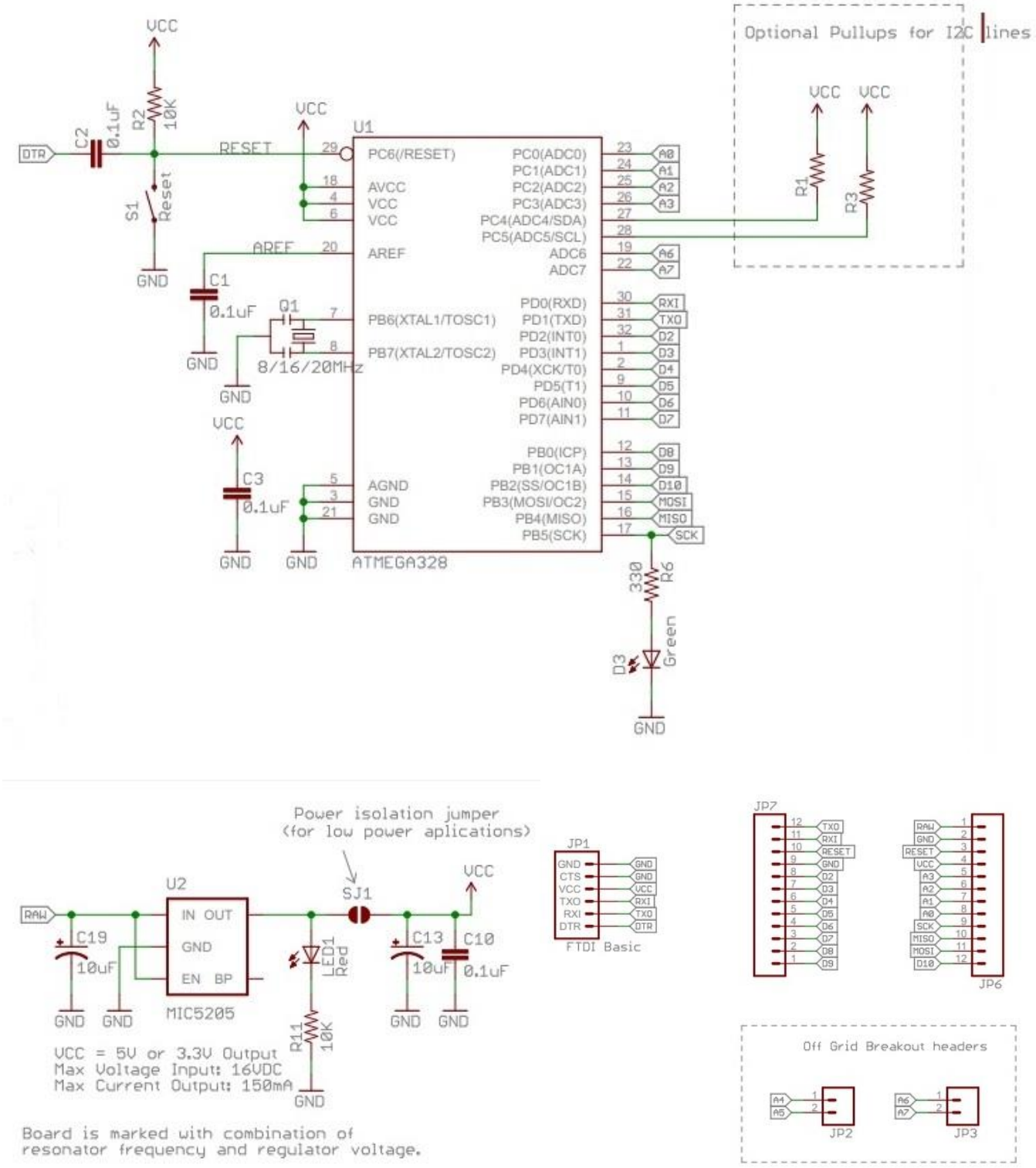


fig. Schematic of arduino pro mini

The Pro Mini's pins surround three of the four sides. The pins on the short side are used for programming, they match up to the FTDI Basic Breakout. The pins on the other two sides are an assortment of power and GPIO pins (just like the standard Arduino).

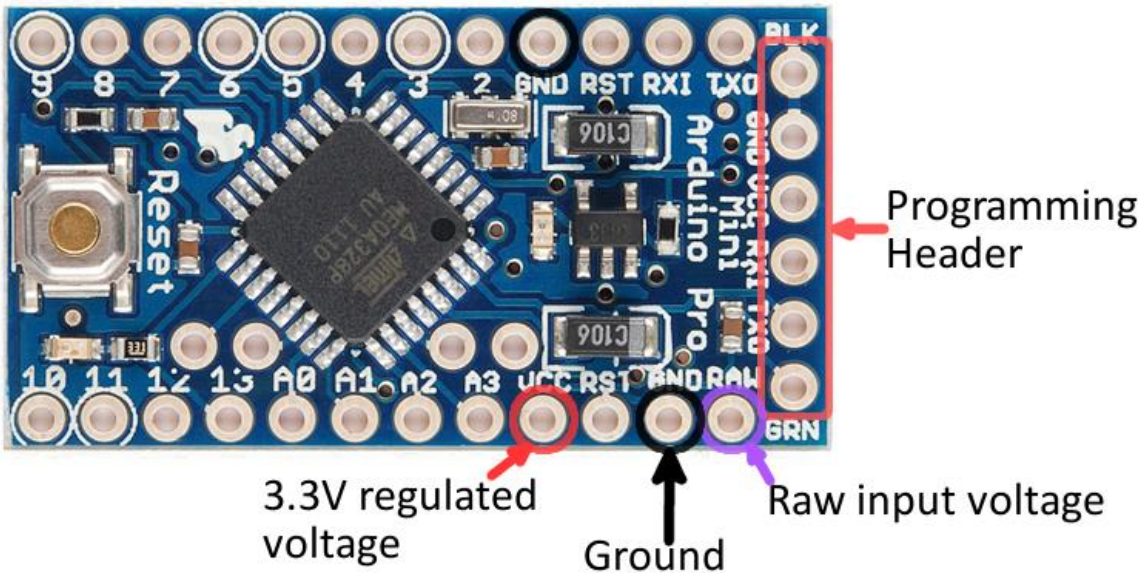


fig. power & programming pins

There are three different power-related pins: GND, VCC, and RAW. GND, obviously, is the common/ground/0V reference. RAW is the input voltage that runs into the regulator. The voltage at this input can be anywhere from 3.4 to 12V. The voltage at VCC is supplied directly to the Pro Mini, so any voltage applied to that pin should already be regulated to 3.3V/5V.

Four pins are actually not located on the edge of the board: A4, A5, A6 and A7. Each of these analog pins is labeled on the back side of the board.

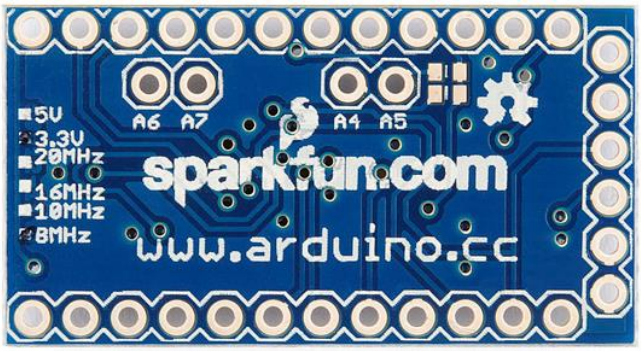


fig. backside of arduino pro mini

A4 and A5's location may be very important if you plan on using I^2C with the Pro Mini – those are the hardware SDA and SCL pins.

More information about pins can be found in our graphical datasheet.

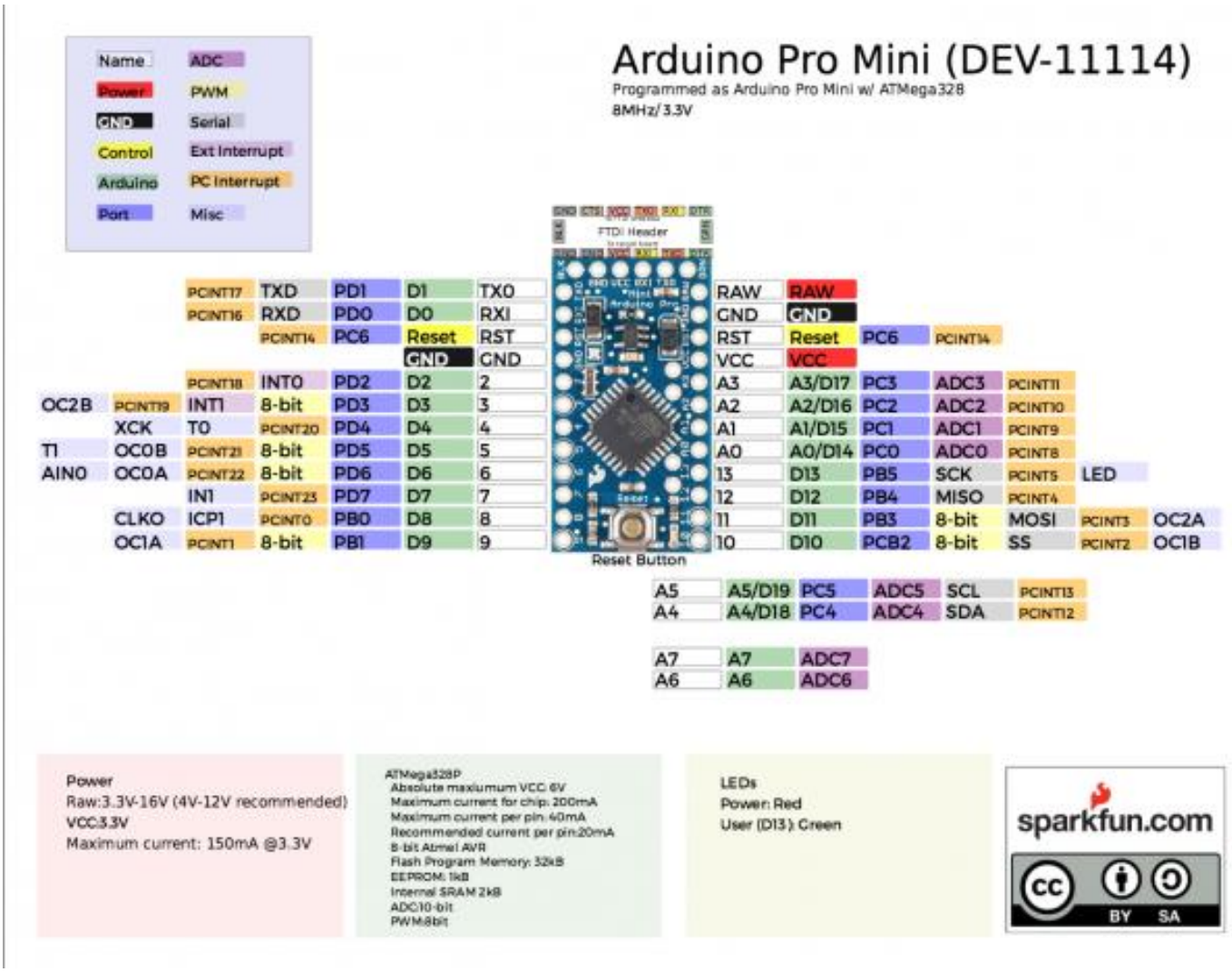


fig. detailed pinout of arduino pro mini

Assembly

The Arduino Pro Mini doesn't look like much when you first get it; it's as bare-bones as can be. We've left it up to you to solder headers or wires into the open through-holes. There are a few things to make you aware of though.

First, decide how you want to connect the FTDI Basic Breakout to the Pro Mini's programming header. The programming header is a row of six pins on the side of the board, labeled "BLK", "GND", "VCC", "RXI", "TXO", and "GRN". Since the FTDI Basic board is equipped with a female header, it's usually best to equip your Mini's programming header with mating male headers, either straight or right-angle.

The remaining assembly choices are up to you. There are many options; you could solder in male headers to make it breadboard-compatible, female headers to make it compatible with jumper wires, or just solder stranded-wire straight into the pins.

Versatility is what makes this board so great, and you can assemble it in whatever way makes the most sense for your project.

Powering

The most important factor in any project is what's going to power it. The Pro Mini doesn't have a barrel jack, or any other obvious way to connect a power supply, so how do you power the thing?

Pick a power source that suits your project. If you want something that matches the compactness of the Pro Mini, a battery – LiPo, alkaline, coin cell, etc. – may be a good choice. Or you could use a wall power supply along with a barrel jack adapter.

If you have a supply that's greater than 3.3V (but less than 12V), you'll want to connect that to the RAW pin on the Mini. This pin is akin to the VIN pin, or even the barrel jack, on the Arduino Uno. The voltage applied here is regulated to 3.3V before it gets to the processor.

If you already have a regulated 3.3V/5V source from somewhere else in your project, you can connect that directly to the VCC pin. This will bypass the regulator and directly power the ATmega328. Don't forget to connect the grounds (GND) too!

There is a third power option that's only usually available while you're programming the Pro Mini. The FTDI Basic Breakout can be used to power the Mini via your computer's USB port. Keep in mind that this option may not be available when your project has entered the wild, absent from any computers or USB supplies.

fig. arduino pro mini connection to PC

That leads us to the next section…programming the Arduino Pro Mini.

Programming

If you've never used an Arduino before (how bold of you to go straight for the Mini!), you'll need to download the IDE. Check out our tutorial on installing Arduino for help on that subject.

The first time you plug the FTDI Basic Breakout in, you may need to install drivers for it. Check out our Installing FTDI Drivers tutorial for help there.

Once both Arduino and the FTDI drivers are installed, it's time to get programming. We'll start by uploading everyone's favorite sketch: Blink. Open up Arduino, then open the Blink sketch by going to *File > Examples > 01.Basics > Blink*.

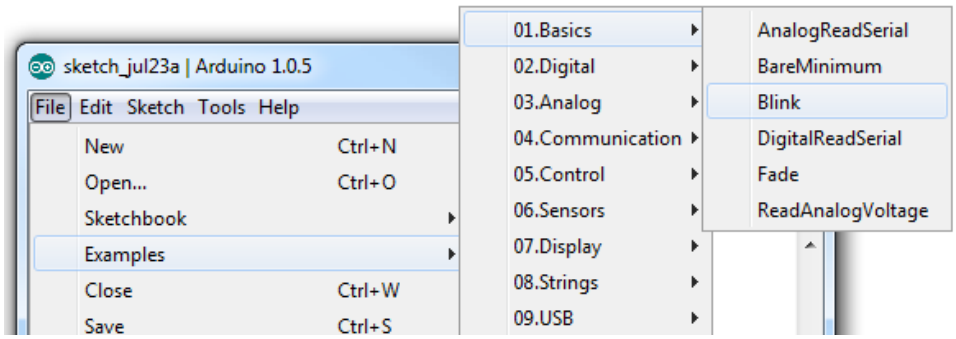


fig. selecting examples in arduino IDE

Before we can upload the sketch to the Mini, you'll need to tell Arduino what board you're using. Go to *Tools > Board* and select Arduino Pro or Pro Mini.

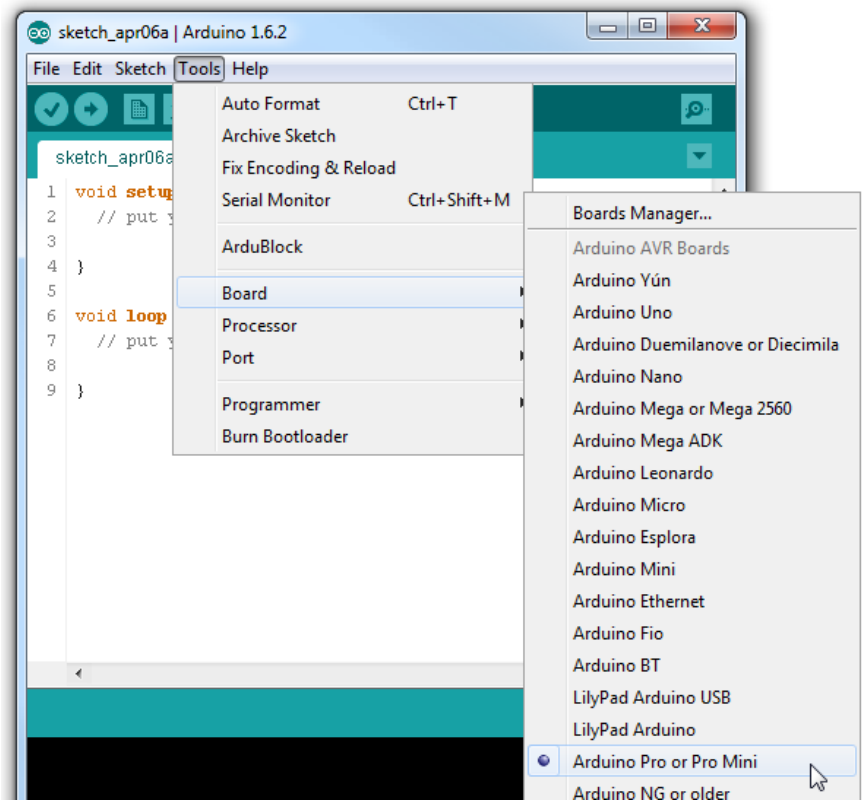


fig. selecting board in arduino IDE

Then, go back up to Tools > Processor and select ATmega328 (3.3V, 8MHz/5V, 16MHz). This tells Arduino to compile the code with an 8MHz/16MHz clock speed in mind, that way the delay(1000); calls will actually delay one second.

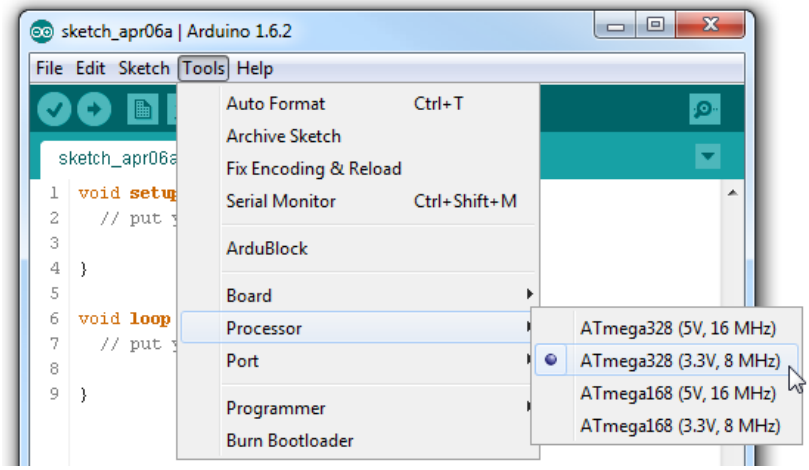


fig. selecting processor in arduino IDE

You'll next need to tell Arduino which serial port your FTDI Basic Breakout has been assigned to. On Windows this will be something like COM2, COM3, etc. On Mac it'll look something like */dev/tty.usbserial-A6006hSc*.

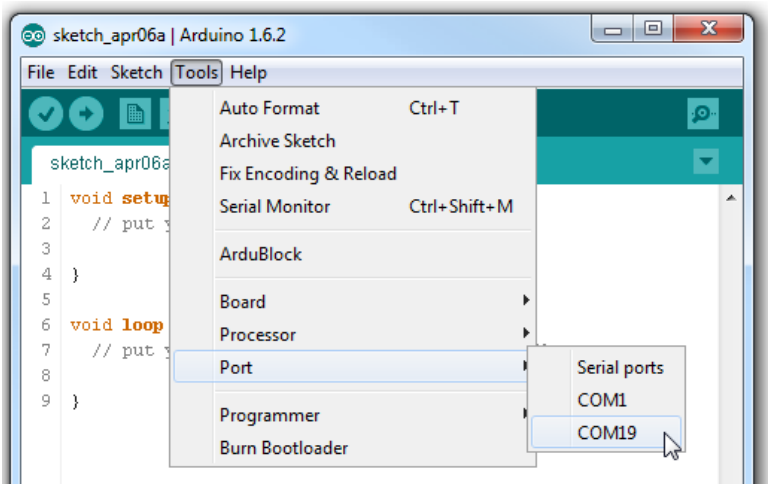


fig. selecting port in arduino IDE

Finally, you're all set to upload the sketch to your Mini. Click on the Upload button (the right-pointing arrow). After a few moments you should see the red and green RX/TX LEDs on your FTDI board flash, followed by a "Done Uploading" message in Arduino's status bar. Voilà, Blinky! The Mini may be missing a few components, but it's got the most important component: LEDs!

2. nRF24L01 Transceiver IC module

The nRF24L01 is a highly integrated, ultra low power (ULP) 2Mbps RF transceiver IC for the 2.4GHz ISM (Industrial, Scientific and Medical) band. With peak RX/TX currents lower than 14mA, a sub µA power down mode, advanced power management, and a 1.9 to 3.6V supply range, the nRF24L01 provides a true ULP solution enabling months to years of battery lifetime when running on coin cells or AA/AAA batteries. The Enhanced ShockBurst™ hardware protocol accelerator additionally offloads time critical protocol functions from the application microcontroller enabling the implementation of advanced and robust wireless connectivity with low cost 3rd-party microcontrollers.

The nRF24L01 integrates a complete 2.4GHz RF transceiver, RF synthesizer, and baseband logic including the Enhanced ShockBurst™ hardware protocol accelerator supporting a high-speed SPI interface for the application controller. No external loop filter, resonators, or VCO varactor diodes are required, only a low cost ±60ppm crystal, matching circuitry, and antenna.

The nRF24L01 is available in a compact 20-pin 4 x 4mm QFN package.

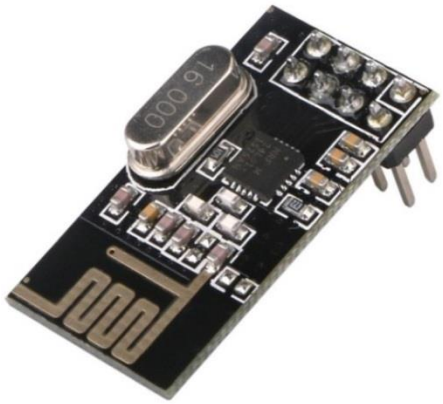


fig. nRF24L01 trans-receiver IC module

Features

- Low cost single-chip 2.4GHz GFSK RF transceiver IC
- Worldwide license-free 2.4GHz ISM band operation
- 1Mbps and 2Mbps on-air data-rate
- Enhanced ShockBurst™ hardware protocol accelerator
- Ultra low power consumption – months to years of battery lifetime
- On-air compatible with all nRF24L Series in 1 and 2Mbps mode
- On-air compatible with nRF24E and nRF240 Series in 1Mbps mode

Specification

The Nordic nRF24L01 is a highly integrated, ultra low power (ULP) 2Mbps RF transceiver IC for 2.4GHz ISM band operation.

Ultra low power consumption

- 900nA deep sleep mode
- 11.3mA Radio TX at 0dBm
- 12.3mA Radio RX at 2Mbps on-air data-rate

2.4GHz Radio

- 2.4GHz ISM band operation
- GFSK modulation, 1 or 2MHz bandwidth
- 0, -6, -12, and -18dBm programmable TX output power
- Configurable on-air data rate of 1Mbps or 2Mbps
- -82dBm RX sensitivity at 2Mbps
- -85dBm RX sensitivity at 1Mbps
- Compatible with a 16MHz ±60ppm crystal

Enhanced ShockBurst™ hardware protocol accelerator

- Automatic packet assembly
  (Preamble, Address, CRC)
- Automatic packed detection and validation
- Dynamic payload length, 1 to 32Bytes
- Auto retransmit
- Auto Acknowledgment with optional payload
- 6 data pipe MultiCeiver™
- 3 separate 32Byte TX and RX FIFOs

MCU Interface

- SPI, up to 10Mbps

Power supply

- Internal linear voltage regulator
- 1.9 to 3.6V supply range

Temperature range of -40 to +80 ºC

Package options

- 20-pin 4 x 4mm QFN

More details and operating principles of nRF24L01 with arduino board

Having two or more Arduinos be able to communicate with each other wirelessly over a distance opens lots of possibilities:

- Remote sensors for temperature, pressure, alarms, much more
- Robot control and monitoring from 50 feet to 2000 feet distances
- Remote control and monitoring of nearby or neighborhood buildings
- Autonomous vehicles of all kinds

These are a series of low-cost 2.4 GHz Radio modules that are all based on the Nordic Semiconductor nRF24L01+ chip. The Nordic nRF24L01+ integrates a complete 2.4GHz RF transceiver, RF synthesizer, and baseband logic including the Enhanced ShockBurst™ hardware protocol accelerator supporting a high-speed SPI interface for the application controller. The low-power short-range (50-200 feet or so)Transceiver is available on a board with Arduino interface and built-in Antenna for less than $3! There is also a high-power version (below).

fig. nRF24L01 high power version

NOTE! Power Problems:

Many users have had trouble getting the nRF24L01 modules to work. Many times the problem is that the 3.3V Power to the module does not have enough current capability, or current surges cause problems. Here are suggestions:

- Use the RF24 Library from TMRH20 (below) and set power low to minimize power requirements:radio.setPALevel(RF24_PA_MIN); Space the two radios about a meter apart. After you have things working, and if you know you have enough 3.3V current (up to 250 mA or more) then try higher power. The possibilities are: RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH and RF24_PA_MAX
- Connect a 3.3 uF to 10 uF (MicroFarad) capacitor directly on the module from +3.3V to Gnd (Watch + and - !) [Some users say 10 uF in parallel with 0.1uF is best] This is especially important if you are connecting the module with jumper wires. Or you are using the regular Arduino UNO which provides only 50 mA at 3.3V
- Use a YourDuinoRoboREDArduino UNO compatible, which has an added 3.3V regulator (But add a .1 uF capacitor on the radio module).

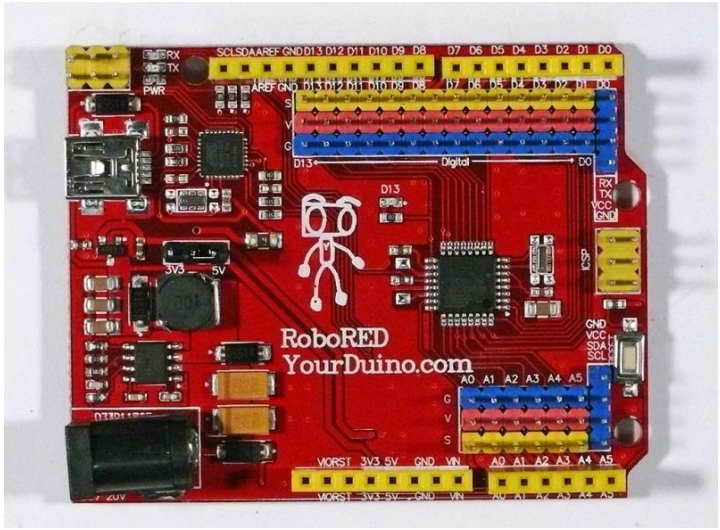


fig.YourDuinoRoboRED

- There are also nice low-cost base modules like THIS(RIGHT) that you can plug nRF24L01 modules into. These have a 3.3V regulator built in, AND good power bypass capacitors. You can also find these on Ebay etc. They make it easier to get started and keep operation reliabl

fig. Base module for nRF24L01

- A separate 3.3V power supply.
- If you design a printed circuit board that the module plugs into, add .1uf and 10uf capacitors close to the GND and 3.3V pins. See the schematic of the base board shown on the right on this page (http://yourduino.com/sunshop//index.php?l=product_detail&p=467) as an example.

These instability problems are particularly noticeable when 3.3V power comes from a UNO, MEGA, Nano etc. that has only 50 ma of 3.3V power available. Newer boards like the YourDuinoRoboRED have 350 mA or more available and can run even the high-power modules directly.

Range

Range is very dependent on the situation and is much more with clear line of sight outdoors than indoors with effects of walls and materials. The usual distance quoted by different suppliers for the low-power version module with the single chip is 200 Feet or 100 Meters. This is for open spaaaaaace between units operating at a Data Rate of 250KHz. Indoors the range will be less due to walls etc... The example with radio.setPALevel(RF24_PA_LOW); will be only 10 feet or so. But reliable. You can do these things to get better range:

1. Make sure you have good 3.3V power (not just plain UNO, Mega etc. on USB power). You can use a separate 3.3V supply, or a Base Module powered from 5V that has a 3.3V regulator, or a YourDuino RoboRED.
2. **After** you have **good** 3.3V power, set the RF "Power Amplifier Level" to MAX. Like this:radio.setPALevel(RF24_PA_MAX); Or try intermediate settings first: RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH and RF24_PA_MAX
3. Set radio.setDataRate(RF24_250KBPS); *Fast enough.. Better range*
4. *Set radio.setChannel(108);* 2.508 Ghz - Above most Wifi Channels

We suggest you test two units at your actual locations before making a decision. There are units with an Antenna Preamplifier for the receiver and transmitter power amplifier and external antenna. The range between that type unit and several low-power units will be better than between two low-power units. Every situation is a little different and difficult to get an exact number without actual-tests.

In datasheet pages 7-8-9 (For Overview and Features), and page 39 (MultiCeiver, which allows 6 Arduinos to talk to a Primary Arduino in an organized manner). Fortunately the board-level products we have take care of many of the physical and electrical details and Antenna Impedance Matching etc.,and this library takes care of lots of register initialization and operational details.

There are other types of nRF24L01 modules which add Transmitter power amplifiers and Receiver preamplifiers for longer distances.. up to 1 Km (3000 feet). See them all here.These modules use an external antenna which can be a simple directly-attached one or a cable-connected antenna with more gain or directivity. Here's what some of these look like:

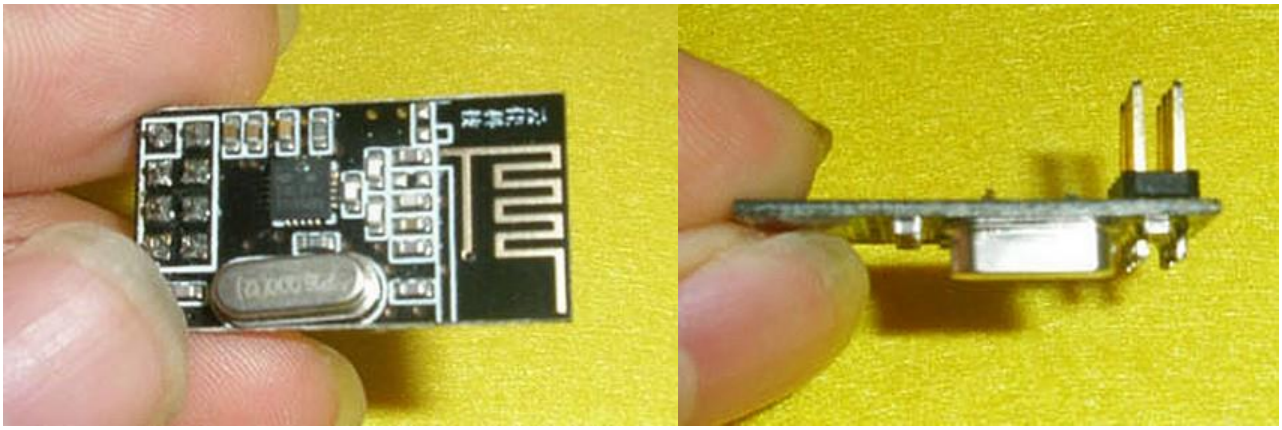


fig.nRF24L01

Above is the low-power version, with it's built-in zig-zag antenna. On the right you can see the pins sticking down (up in this photo) that connect to Arduino. Later we will show the pinout. **NOTE!! Different Libraries use different pinouts from nRF24L01 to Arduino!!**

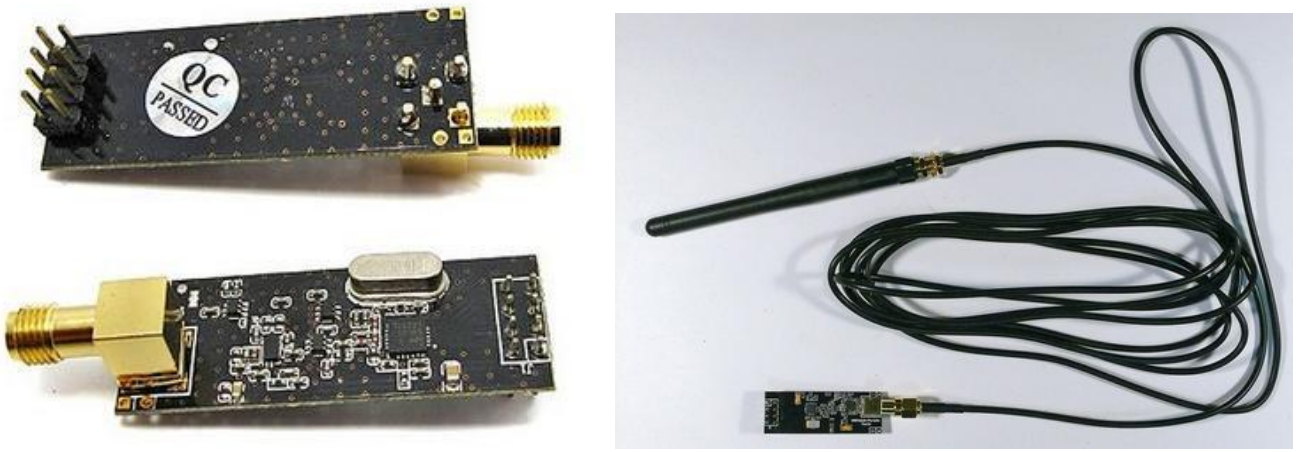


fig. nRF24L01 power module with external antenna

Above is the version with Transmit Power amplifier and Receive Preamplifier. Our low-cost antenna is on the unit in the middle. The same 8 pins connect to Arduino and the base module. Same software is used. On the right is the antenna connected with a 3M cable. Connectors are "Reverse SMA".

These transceivers use the 2.4 GHz unlicensed band like many WiFi routers, BlueTooth, some cordless phones etc. The range is 2.400 to 2.525 Ghz which is 2400 to 2525 MHz (MegaHz). The nRF24L01 channel spacing is 1 Mhz which gives 125 possible channels numbered 0 .. 124. WiFi uses most of the lower channels and we suggest using the highest 25 channels for nRF24L01 projects.

Transceivers like these both send and receive data in 'packets' of several bytes at a time. There is built-in error correction and resending, and it is possible to have one unit communicate with up to 6

other similar units at the same time. The RF24 Network Library extends this to multiple 'layers' of interconnected_transceivers.

These amazing low-cost units have a lot of internal complexity but some talented people have written Arduino libraries that make them easy to us. They all use the same pinout as shown in the following diagram, which is a TOP VIEW (Correction!):

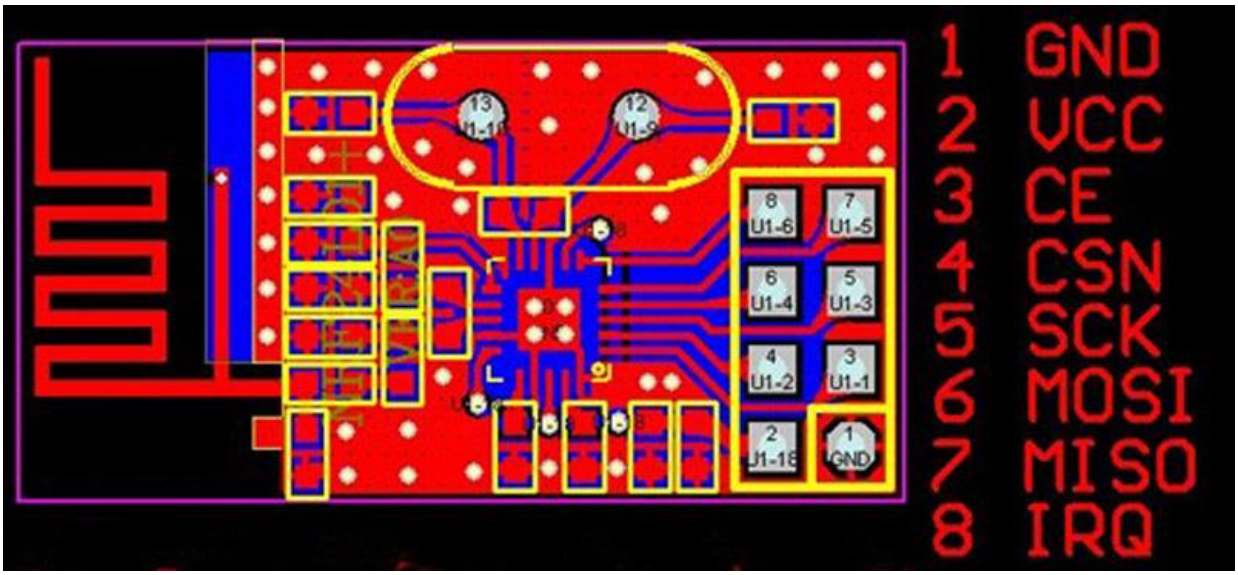


fig. nRF24L01 pcb design

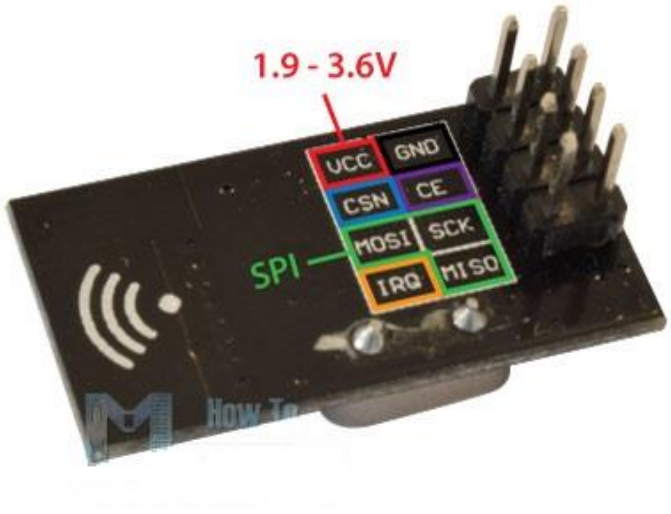


fig. nRF24L01 pinout

Here are details of the Pinout and connections to Arduino (updated):

Signal	RF Module PIN	Cable COLOR	"Base Module" PIN	Arduino pin for TMRh20 RF24 Library	Arduino pin for RF24 Library	MEGA2560 pin RF24 Library	Arduino Pin for RH_NRF24 RadioHead Library	MEGA2560 Pin for RH_NRF24 RadioHead Library
GND	**1**	**Brown**	**GND**	**GND**	**GND ***	**GND ***	**GND ***	**GND ***
VCC	**2**	**Red**	**VCC**	**3.3 V**	**3.3V ***	**3.3V ***	**3.3V ***	**3.3V ***
CE	**3**	**Orange**	**CE**	**7**	**9**	**9**	**8**	**8**
CSN	**4**	**Yellow**	**CSN**	**8**	**10**	**53**	**10**	**53**
SCK	**5**	**Green**	**SCK**	**13**	**13**	**52**	**13**	**52**
MOSI	**6**	**Blue**	**MO**	**11**	**11**	**51**	**11**	**51**
MISO	**7**	**Violet**	**MI**	**12**	**12**	**50**	**12**	**50**
IRQ	**8**	**Gray**	**IRQ**	**-**	**2**	**per library**	**N/C**	**N/C**

- NOTE!! Most * problems with intermittent operation are because of insufficient current or electrical noise on the 3.3V Power supply. The MEGA is more of a problem with this. Solution: ADD bypass capacitors across GND and 3.3V ON the radio modules **or use the Base Modules** shown above. One user said, "Just Solder a 100nF ceramic cap across the gnd and 3.3v pins direct on the nrf24l01+ modules!" Some have used a 1uF to 10uF capacitor.
- NOTE: Pin 8 IRQ is Unused by most software, but the RF24 library has an example that utilizes it.

The COLOR is for optional color-coded flat cable. Photos above show an example.

NOTE: These units VCC connection ***must*** go to 3.3V not 5.0V, although the Arduino itself may run at 5.0V and the signals will be OK. The NRF24L01+ IC is a 3.3V device, but its I/O pins are 5 V tolerant , which makes it easier to interface to Arduino/YourDuino.

Arduino UNO and earlier versions have a 3.3V output that can run the low-power version of these modules (See Power Problems at the top of this page!), but the high-power versions must have a separate 3.3V supply or use a Base Module with a 3.3V regulator. The YourDuino RoboRED has a

higher power 3.3V regulator and can be used to run the high-power Power Amplifier type module without a separate 3.3V regulator.

nRF24L01 SOFTWARE AND LIBRARIES:

We will show an example of transmit and receive software below, and there are many examples on the RF24 Library download page. You will need a library of software to run the nRF24L01 radios. There has been a progression of improvements to the RF24 libraries by many people. TMRH20 (Who IS he/she, anyway??) has done a great job on the current library and network additions. There are lots of details but you can ignore many of them that the library will take care of.

Get TMRh20's excellent RF24 Library that supports both Arduino and RaspberryPi:

NOTE!! If you have an earlier version of an RF24 Library, COMPLETELY REMOVE IT and install this version.

Read about it and Download it (https://github.com/nRF24/RF24) (Click "Download ZIP" on the upper right of the page)
Read the detailed documentation (http://tmrh20.github.io/RF24/)
Read his BLOG with advanced projects (http://tmrh20.blogspot.in/)
Once you have downloaded the ZIP, you should see a folder called RF24-master.ZIP. Change the name of this file to just RF24.ZIP. Double click on the ZIP and you should see a folder inside also called RF24-master. Rename this to just RF24 as well.

Then see our page about installing libraries (http://arduino-info.wikispaces.com/Arduino-Libraries)
When you have the library installed, you can run some examples. A ways below we discuss the individual commands (methods) in the RF24 Library and show how to use them to make a simple transmit-receive system. But to get you started we suggest you start with one of the Examples in the RF24 Library from TMRh20. See this page:

https://github.com/TMRh20/RF24/tree/master/examplesWhen you install the RF24 library, those examples are loaded on your computer. You don't have to download them.

We suggest you start with, um, "GettingStarted". You need to set up two arduinos, with two nRF24L01's like in the photo above. Separate them by 5 feet or so, though. . You can run two instances of the Arduino IDE on the same machine, with two nRF24L01's on separate USBs with separate COM#. I suggest this sequence:

1. Connect one nRF24L01 to your first Arduino according to the table above (Use the TMRh20 RF24 Library column connections).
2. Start the Arduino IDE .. connect the first Arduino with USB. Find the COM number (Tools>Port) and select it.

3. Load the GettingStarted example (<File>Examples>RF24>) and make sure it compiles OK. UPLOAD to your Arduino

4. Open the Serial Monitor (right side if the dark green bar) and SET IT TO 115200 . Close it and open it again. you should see:

```
RF24/examples/GettingStarted
*** PRESS 'T' to begin transmitting to the other node
```

5. Connect the second nRF24L01 to your second Arduino with same connections according to the table above.

6. Start another instance of the Arduino IDE .. connect second Arduino with a second USB cable. Find the COM number (Tools>Port) and select it.

7. Load the example and make sure it compiles OK. UPLOAD to your Arduino

8. Open the Serial Monitor (right side if the dark green bar) and SET IT TO 115200 . Close it and open it again. you should see the Same output.

9. Close the Serial Monitor, look at the code in the second IDE window and CHANGE one line to look like this:

```
/****************** User Config***************************/
/***      Set this radio as radio number 0 or 1         ***/
bool radioNumber = 1;
```

10. Upload that version to the second Arduino.

11. Now, open both Serial Monitor windows, and type "T" in the top of the first Arduino window.

12. You should see some thing like:

```
RF24/examples/GettingStarted
*** PRESS 'T' to begin transmitting to the other node
*** CHANGING TO TRANSMIT ROLE -- PRESS 'R' TO SWITCH BACK
Now sending
Sent 523513060, Got response 523511232, Round-trip delay 1828
microseconds
Now sending
Sent 524519544, Got response 524517728, Round-trip delay 1816
microseconds
```

And the other Serial Port window should look like:

```
RF24/examples/GettingStarted
*** PRESS 'T' to begin transmitting to the other node
Sent response 523511232
Sent response 524517728
Sent response 525521620
```

You can reverse the actions by type R in the transmitting side and then T in the opposite side.

OK? You can always revert to this test when things seem to stop working. NOTE: See the "POWER PROBLEMS" section above if nothing works!

You can run some other examples, but when you want to understand the RF24 library and it's Methods better, see the section below.

See: Other Example Sketches at top of this page. Also see the very good example by Robin on (http://forum.arduino.cc/index.php?topic=421081).

The reader will find more details and additional examples here:

RF24 LIBRARIES: Information, Documentation, Network, Mesh Network and more!

RF24 Library: Main Page. Download ZIP file at right
RF24 Library detailed Documentation
RF24 Library CLASS Documentation (details of a funcions and their valid parameters)
RF24 Library Examples Many examples of the capabilities of the RF24 Library
RF24 Network System A many-node Network Library
RF24 library for mesh networking A Mesh Network Library
RF24 Library for transmitting Audio

RF24 LIBRARIES: Commonly Used Commands (Methods):

NOTE: If you wish, see ALL the details (https://tmrh20.github.io/RF24/classRF24.html)
We will try to understand the commands to set up and operate an nRF24L01 radio and build a very simple working example.

First we will look at the minimum needed to set up and transmit or receive data. Later we will write more complete examples and get them working with real radios..

CREATE AND START UP A RADIO (For either transmit or receive):

RF24 (uint8_t _cepin, uint8_t _cspin) Create a radio and set the Arduino pins to be used for CE and CS)

EXAMPLE: (Create an instance of a radio, specifying the CE and CS pins.)
RF24 **myRadio** (7,8); *"myRadio" is the identifier you will use in the following examples*

NOTE: *The following pins are fixed and unchangeable. AND Vcc (supply voltage MUST be 3.3V)*
SCK to Arduino pin 13
MOSI to Arduino pin 11
MISO to Arduino pin 12

NOTE*: In all following commands, you must use the same identifying name you used in the RF24 statement that created the radio. ("**myRadio**" in the example above) You will use that identifier followed by (dot) followed by the method (command), followed by parameters. This is "object oriented programming". The OBJECT in this case is the Radio, and there are many METHODS that can operate on the Object. A METHOD is much like a Function or a Subroutine.*

EXAMPLE: ***myRadio.begin();*** Start up the actual radio module with the "begin" method

NOTE: "PIPES" : This is often confusing. nRF24L01 uses "pipes" that connect from transmitter to receiver. Pipes have an address you need to set. The Transmitter pipe must have the same address as the Receiver pipe. Later it's possible to use multiple "pipes" at once

---

EXAMPLE OF RECEIVING DATA:

myRadio.openReadingPipe (1, const uint8_t *address) Pipe number (usually 1), pipe address (which is usually 5 bytes in an array structure).

EXAMPLE:
byte addresses[][6] = {"1Node"}; Create address for 1 pipe.
myRadio.openReadingPipe(1, addresses[0]); *Use the first entry in array 'addresses' (Only 1 right now)*

myRadio.startListening *(); Turn on the receiver and listen for received data. You MUST have opened a reading pipe FIRST.*

if(***myRadio.available****())* Check for available incoming data from transmitter
{
while **(myRadio.available**()) *While there is data ready*
{

myRadio.read*(&myData, sizeof(myData)) ;* Get the data payload (You must have defined that already!)
}
myRadio.stopListening(); *stop listening*

EXAMPLE OF TRANSMITTING DATA:
byte addresses[][6] = {"1Node"}; Create address for 1 pipe.
myRadio.openWritingPipe(1, addresses[0]); *Use the first entry in array 'addresses' (Only 1 right now)*

myRadio.write*(* &myData, sizeof(myData) *)*

We have written working examples of Transmit and Receive sketches using these methods.(http://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-ExampleSketches#bm1)

SET SOME OTHER OPERATING OPTIONS AND PARAMETERS:

NOTE: Many of these methods have default values you can usually use.

radio.printDetails();
Prints out a LOT of debugging information.

---

radio.setDataRate(RF24_250KBPS);
speed RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or RF24_2MBPS for 2Mbps. 250K Bits per second gives longest range.

---

radio.setPALevel(RF24_PA_MAX);
Set Power Amplifier (PA) level to one of four levels: RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH and RF24_PA_MAX
The power levels correspond to the following output levels respectively: NRF24L01: -18dBm, -12dBm,-6dBM, and 0dBm
0 dBm is equal to 1 milliwatt. Each 3 dB is a 2:1 ratio. 6 dB is 4:1 10 dB is 10:1 So -18 dBm is 0.0000158489 watts !
To calculate all this dBm Stuff (Which us old Radio Engineers love)
See (https://www.rapidtables.com/convert/power/dBm_to_Watt.html)
The "High Power" nRF24L01 modules have a gain of 100, so their output is +20 dBm (100 milliwatts)

---

radio.setChannel(108);

Which RF channel to communicate on, 0-124 Can operate on frequencies from 2.400GHz to 2.524GHz. Programming resolution of channel frequency is 1Mhz
This is the same unlicensed band WiFi operates in (WiFi uses 2.400 to 2.500 gHz). Usually frequencies above channel 100 are best.
NOTE: In most countries the allowed frequencies are from 2.400GHz to 2.483.5GHz which you must not exceed. In USA it's best to use channels from 70 to 80.

You can scan the channels in your environment to find a channel that is clear... See scanner scatch. (http://arduino-info.wikispaces.com/Nrf24L01-Poor%20Man%27s%202.4%20GHz%20Scanner).

nRF24L01 Intermittent / No operation.

ALWAYS check 3.3V power. Many times intermittent operation is due to power supply regulation issues. Even though the average current may be less than 15ma, apparently there are quick transients when each transmit burst happens. The first examples in the RF24 TMRH20 library run low power and this is less of a problem.
I used to have nRF24L01 problems. Worked one day / one minute, failed the next. Now I put a 0.1uf AND 10uf capacitor right from GND to 3.3V pins on the modules, and things are MUCH better. An excellent solution is the base module shown at the top of this page.

These modules are designed to plug into a baseboard that usually has significant bypass capacitors. If you have them on the end of jumper wires you need extra bypass capacitors.

1st Module

In our IoT (Internet of Things) based home automation system project , this module is our main module. Because, this module receives all the data from other 3 modules , send those data to the database and the database send back those data to our mobiles.

This module does multiple jobs at the same time. The circuitry is very complex and hard to trouble-shoot, because various segments of electronics (power, communication) and software technologies are involved here.

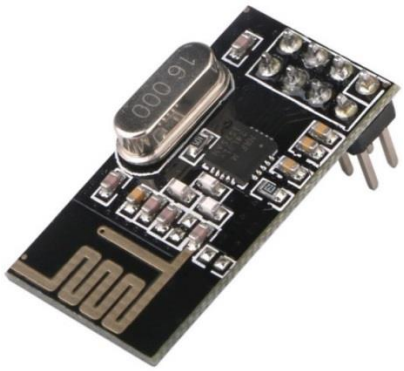
So, the brain of the module is ESP-12E designed by AI-Thinker. It is a Wifi module consists of the ESP8266 MCU chip. There are many other packages available in the market , but, ESP-12E is very popular among them. ESP-12E doesn't have so much data communication ports , so we used the other module, Arduino nano which actually assists the ESP-12E wifi module. the communication between this and other modules is done by nRF24L01 receiver module.

OBJECTIVE

- Control (ON-OFF, speed control of fan/ brightness control of light bulb) AC (Alternate Current) appliances, ex. fan and bulb .
- Collect and process data from other 3 modules.
- Send those collected data to the database via internet.

PLANNING OF COMPONENTS

- **nRF24L01 Module** – This is the module where the data from other modules are first received and then the data is sent to Arduino.



- **Arduino Nano --** Act as a helping hand of ESP12E , receive all the data from other 3 modules (gas detection module, pir motion sensing module & temperature sensing module) and send it to the ESP12E via Tx-Rx channel for further processing.

- **ESP12E Wi-Fi module** – Transmit, receive & process all data from other modules and control the ac appliances directly.

- **Relay –** 5V Relays are used to ON-OFF the appliances.

- **BT136** – Triac for switching and controlling appliances.

- **MOC3020M** -- Random-phase triac driver output optocoupler (250/400 Volt Peak).

- **MCT2E** – Optocoupler, phototransistor output with base connection

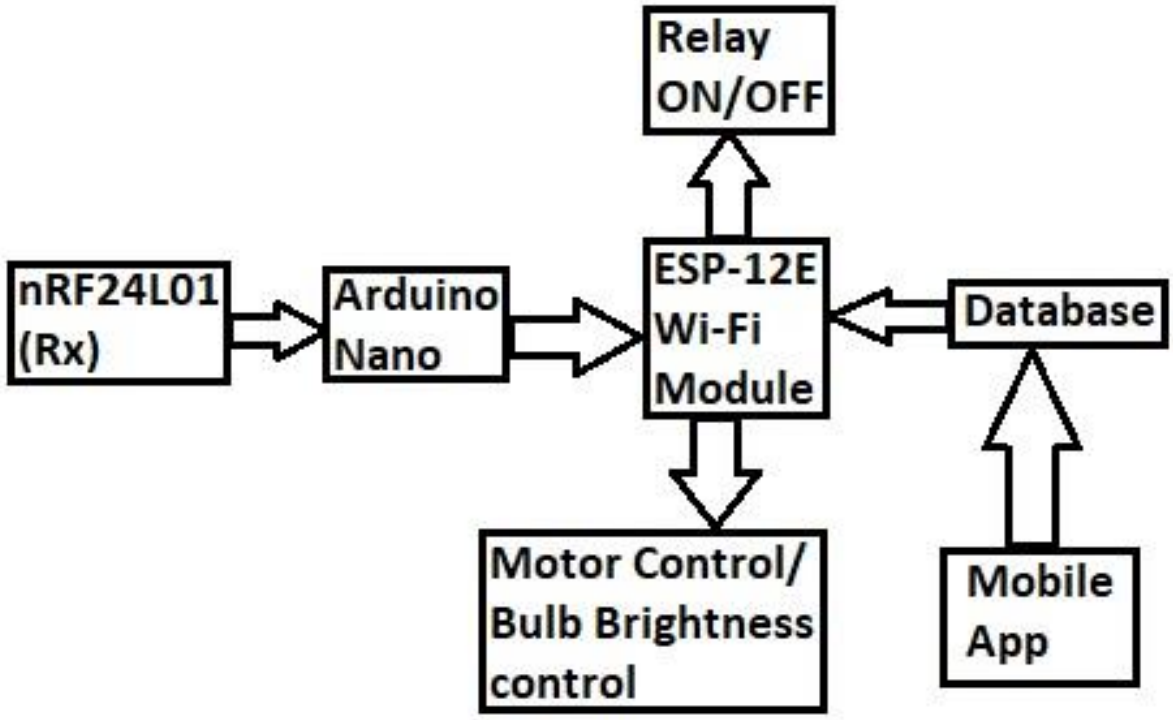
- **PIC12F675** -- Flash-based 8 bit CMOS mcu for for Triac for controlling motor/bulb precisely.

- BC557 and 1N4004 diodes are used as relay driver.

- 120KΩ resistance and 0.1µF capacitors are used as a filter for smooth motor controlling.

- BC547 is used to amplify the received data signal from Arduino Nano.

BLOCK DIAGRAM



MAJOR COMPONENTS DESCRIPTION

Arduino Nano

overview

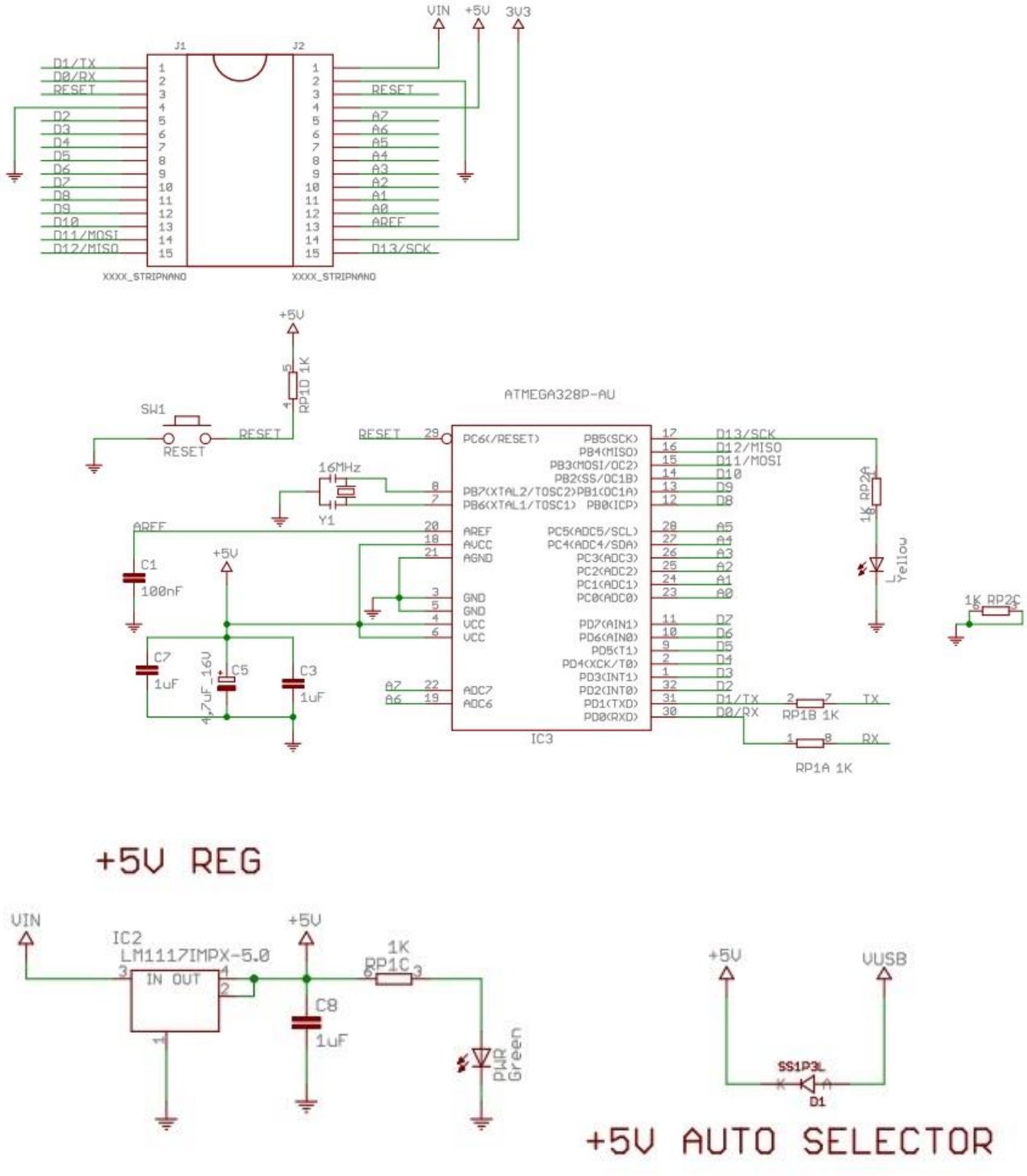
The Arduino Nano is a compact board similar to the UNO. The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

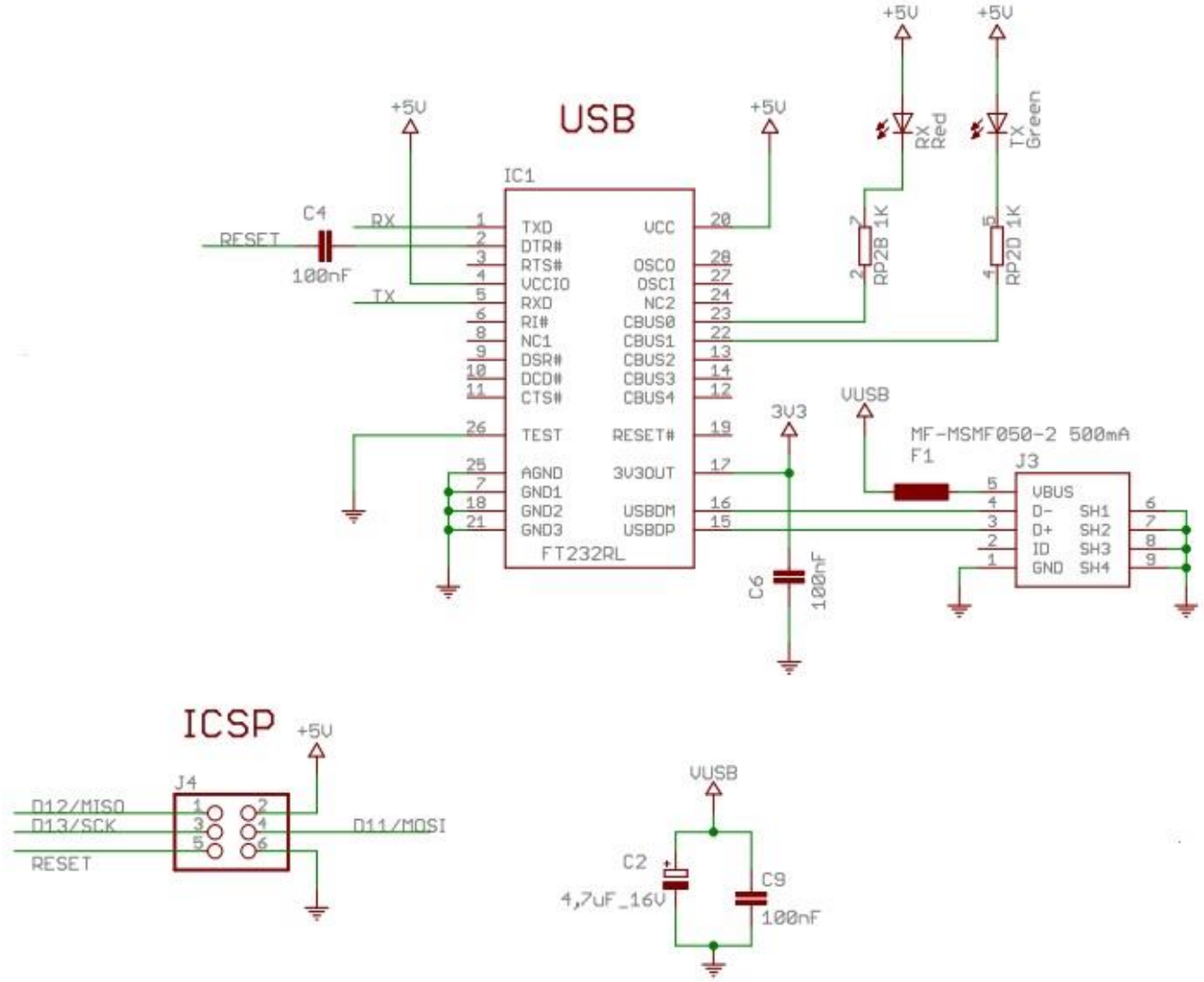
fig. Arduino Nano

Tech Specs

Microcontroller	ATmega328p
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12 V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005

Schematic Diagram





Power

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

Memory

The ATmega328P has 32 KB, (also with 2 KB used for the bootloader. The ATmega328P has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano 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 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- 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.

The Nano has 8 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 analogReference() function. Analog pins 6 and 7 cannot be used as digital pins. Additionally, some pins have specialized functionality:

- I2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

There are a couple of other pins on the board:

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

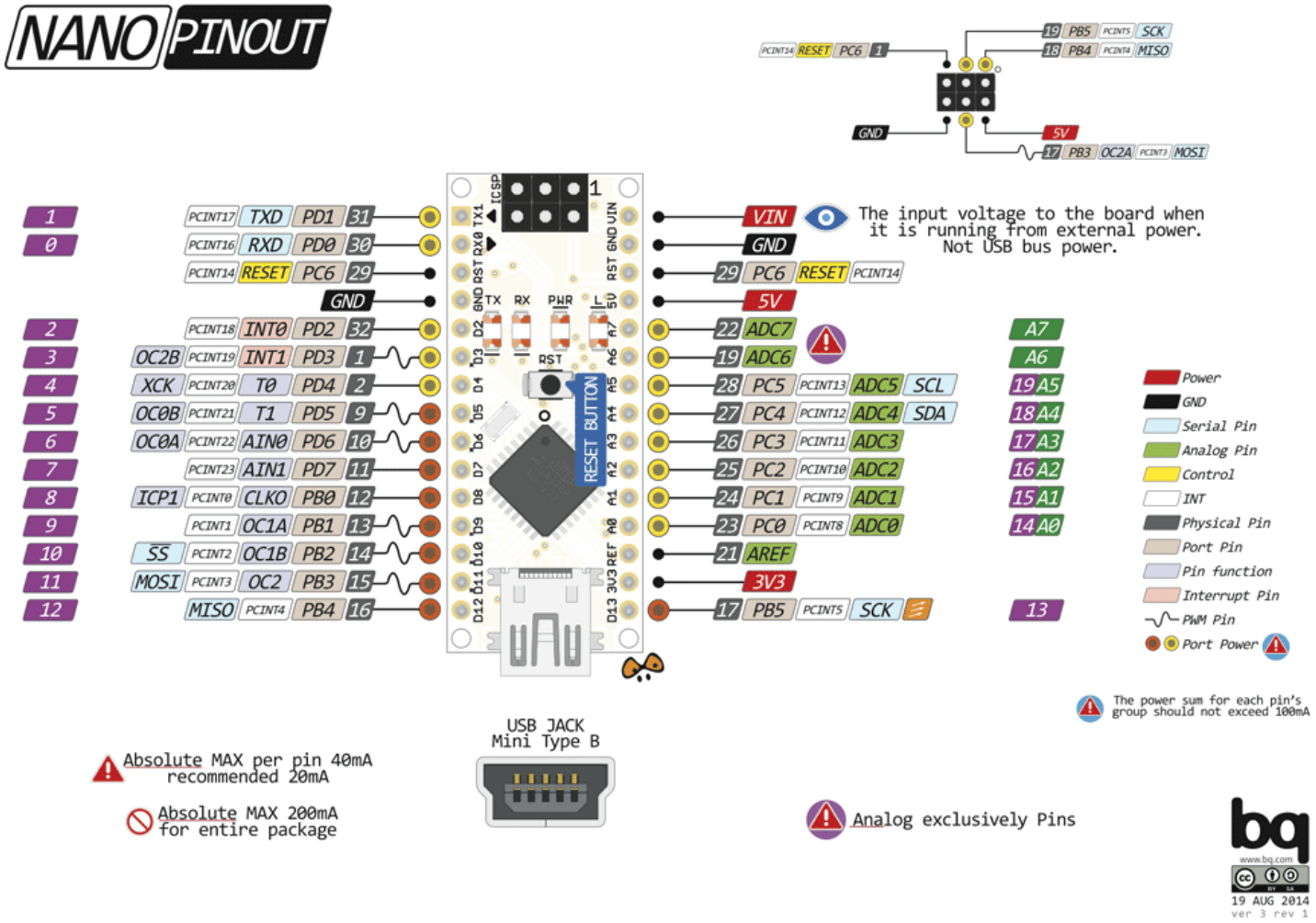


fig. Arduino Nano pinout

Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328P provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Nano's digital pins. The ATmega328P also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. To use the SPI communication, please see ATmega328P datasheet.

Programming

The Arduino Nano can be programmed with the Arduino software. Select "Arduino Duemilanove or Nano w/ ATmega328P" from the Tools > Board menu (according to the microcontroller on your board). The ATmega328P on the Arduino Nano 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. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Nano 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 FT232RL is connected to the reset line of the ATmega328P 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 Nano 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 Nano. 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.

ESP-12E Wi-Fi Module

Preambles

The ESP-12E is a wi-fi module consists of ESP8266 which is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer, Espressif Systems and the wi-fi module is designed and produced by Ai-thinker. The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer, Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at the time there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.

The ESP8285 is an ESP8266 with 1 MB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi.The successor to these microcontroller chips is the ESP32. ESP-12E WiFi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Tensilica L106 integrates industry-leading ultra low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna.

The module supports standard IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the add modules to an existing device networking, or building a separate network controller. ESP8266 is high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

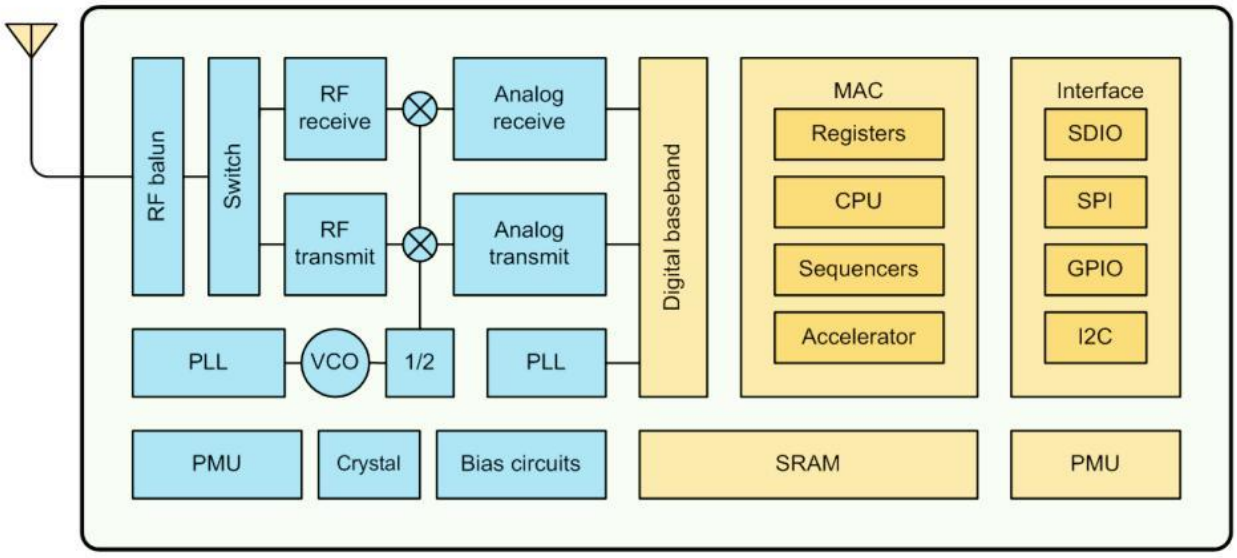


fig. ESP8266EX Block Diagram

ESP8266EX offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor.

When ESP8266EX hosts the application, it boots up directly from an external flash. In has integrated cache to improve the performance of the system in such applications. Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any micro controller based design with simple connectivity (SPI/SDIO or I2C/UART interface).

ESP8266EX is among the most integrated WiFi chip in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the Wi-Fi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs; codes for such applications are provided in examples in the SDK. Espressif Systems' Smart Connectivity Platform (ESCP) demonstrates sophisticated system-level features include fast sleep/wake context switching for energy-efficient VoIP, adaptive radio biasing. for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

Features

• 802.11 b/g/n

• Integrated low power 32-bit MCU

• Integrated 10-bit ADC

• Integrated TCP/IP protocol stack

• Integrated TR switch, balun, LNA, power amplifier and matching network

• Integrated PLL, regulators, and power management units

• Supports antenna diversity

• Wi-Fi 2.4 GHz, support WPA/WPA2

• Support STA/AP/STA+AP operation modes

• Support Smart Link Function for both Android and iOS devices

• Support Smart Link Function for both Android and iOS devices

• SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO

• STBC, 1x1 MIMO, 2x1 MIMO

• A-MPDU & A-MSDU aggregation and 0.4s guard interval

• Deep sleep power < 5uA

• Wake up and transmit packets in < 2ms

• Standby power consumption of < 1.0mW (DTIM3)

• +20dBm output power in 802.11b mode

• Operating temperature range -40C ~ 125C

Parameters

Table 1 below describes the major parameters.

Table 1 Parameters

Categories	Items	Values
WiFi Paramters	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Hardware Paramaters	Peripheral Bus	UART/HSPI/I2C/I2S/Ir Remote Contorl
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	16mm*24mm*3mm
	External Interface	N/A
Software Parameters	Wi-Fi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host
	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Pin Descriptions

There are altogether 22 pin counts, the definitions of which are described in Table 2 below.

Table 2 ESP-12E Pin design

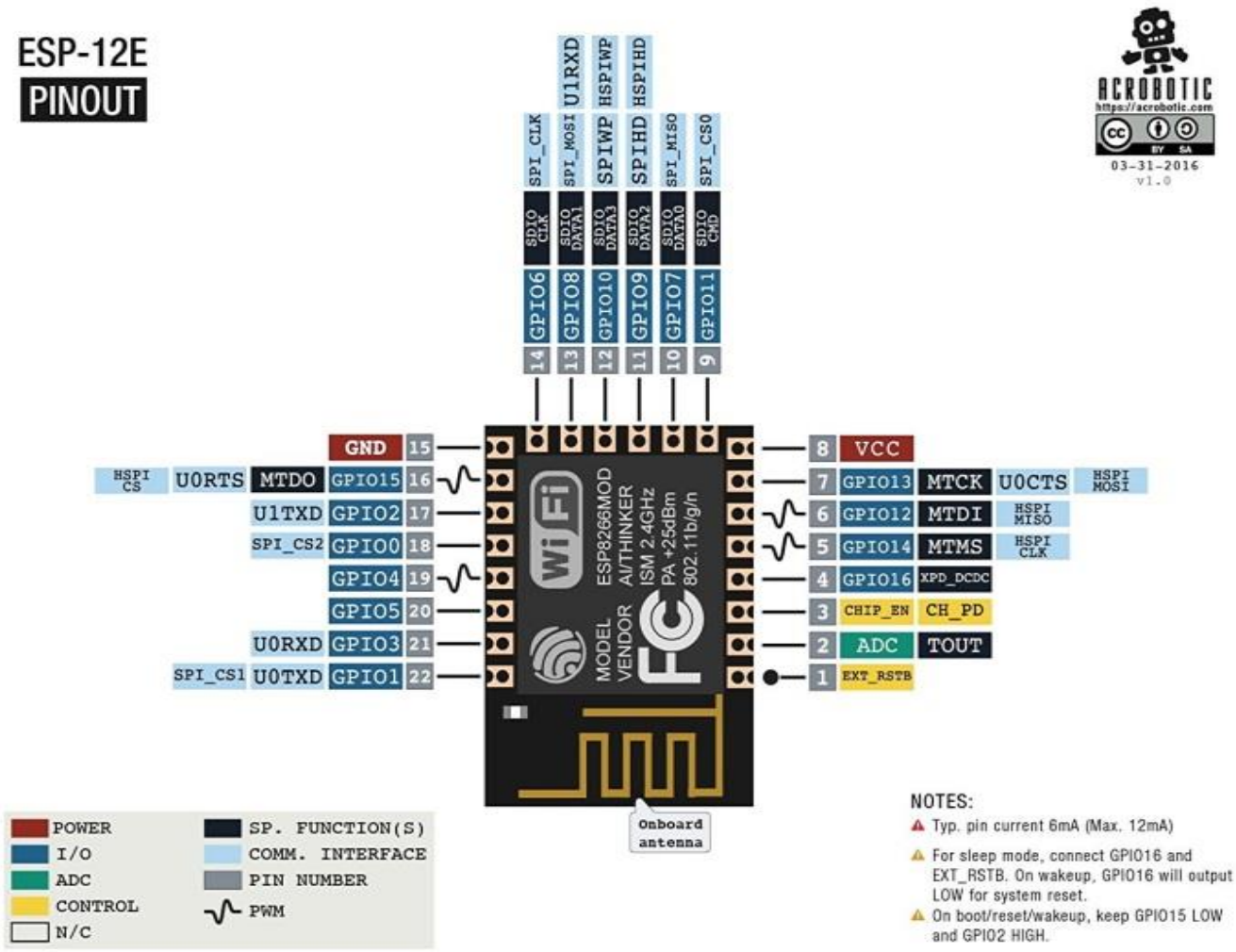


Table 3 Pin Descriptions

NO.	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result.Input voltage range 0-1v,scope:0-1024
3	EN	Chip enable pin.Active high
4	IO16	GPIO16; can be used to wake up the chipset from deep sleep mode.
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input
11	IO9	GPIO9
12	IO10	GBIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	IO2	GPIO2; UART1_TXD
18	IO0	GPIO0
19	IO4	GPIO4
20	IO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

Table 4 Pin Mode

Mode	GPIO15	GPIO0	GPIO2
UART	Low	Low	High
Flash Boot	Low	High	High

Packaging and Dimension

The external size of the module is 16mm*24mm*3mm, as is illustrated in Figure below. The type of flash integrated in this module is an SPI flash, the capacity of which is 4 MB, and the package size of which is SOP-210mil. The antennaapplied on this module is a 3DBi PCB-on-board antenna.

*fig. [Module Pin Counts, 22 pin, 16 mm *24 mm *3 mm]*

Table 5 Dimension of ESP-12E WiFi Module

Length	Width	Height	PAD Size(Bottom)	Pin Pitch
16 mm	24mm	3 mm	0.9 mm x 1.7 mm	2mm

Functional Descriptions

1. MCU

ESP8266EX is embedded with Tensilica L106 32-bit micro controller (MCU), which features extra low powerconsumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz.ESP8266EX is often integrated with external sensors and other specific devices through its GPIOs; codes for such applications are provided in examples in the SDK.

2. Memory Organization

2.1. Internal SRAM and ROM

ESP8266EX WiFi SoC is embedded with memory controller, including SRAM and ROM. MCU can visit the memoryunits through iBus, dBus, and AHB interfaces. All memory units can be visited upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK provided, SRAM space that is available to users is assigned as below:

▪RAM size < 36kB, that is to say, when ESP8266EX is working under the station mode and is connected to therouter, programmable space accessible to user in heap and data section is around 36kB.)

▪ There is no programmable ROM in the SoC, therefore, user program must be stored in an external SPI flash.

2.2. External SPI Flash

This module is mounted with an 4 MB external SPI flash to store user programs. If larger definable storage space is required, a SPI flash with larger memory size is preferred. Theoretically speaking, up to 16 MB memory capacity can be supported.

Suggested SPI Flash memory capacity:

▪OTA is disabled: the minimum flash memory that can be supported is 512 kB;

▪OTA is enabled: the minimum flash memory that can be supported is 1 MB.

Several SPI modes can be supported, including Standard SPI, Dual SPI, and Quad SPI.

Therefore, please choose the correct SPI mode when you are downloading into the flash, otherwise firmwares/programs that you downloaded may not work in the right way.

3. Crystal

Currently, the frequency of crystal oscillators supported include 40MHz, 26MHz and 24MHz. The accuracy ofcrystal oscillators applied should be ±10PPM, and the operating temperature range should be between -20°C and 85°C.

When using the downloading tools, please remember to select the right crystal oscillator type. In circuit design, capacitors C1 and C2, which are connected to the earth, are added to the input and output terminals of the crystal oscillator respectively. The values of the two capacitors can be flexible, ranging from 6pF to 22pF, however, the specific capacitive values of C1 and C2 depend on further testing and adjustment on the overall performance of the whole circuit. Normally, the capacitive values of C1 and C2 are within 10pF if the crystal oscillator frequency is 26MHz, while the values of C1 and C2 are 10pF<C1, C2<22pF if the crystal oscillator frequency is 40MHz.

4. Interfaces

Table 6 Descriptions of Interfaces

Interface	Pin Name	Description
HSPI	IO12(MISO) IO13(MOSI) IO14(CLK) IO15(CS)	SPI Flash 2, display screen, and MCU can be connected using HSPI interface.
PWM	IO12(R) IO15(G) IO13(B)	Currently the PWM interface has four channels, but users can extend the channels according to their own needs. PWM interface can be used to control LED lights, buzzers, relays, electronic machines, and so on.
IR Remote Control	IO14(IR_T) IO5(IR_R)	The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are used by this interface. The frequency of modulated carrier signal is 38KHz.
ADC	TOUT	ESP8266EX integrates a 10-bit analog ADC. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously. This interface is typically used in sensor products.
I2C	IO14(SCL) IO2(SDA)	I2C interface can be used to connect external sensor products and display screens, etc.

Interface	Pin Name	Description
UART	**UART0:** TXD (U0TXD) RXD (U0RXD) IO15 (RTS) IO13 (CTS) **UART1:** IO2(TXD)	Devices with UART interfaces can be connected with the module. Downloading: U0TXD+U0RXD or GPIO2+U0RXD Communicating: UART0: U0TXD, U0RXD, MTDO (U0RTS), MTCK (U0CTS) Debugging: UART1_TXD (GPIO2) can be used to print debugging information. By default, UART0 will output some printed information when the device is powered on and is booting up. If this issue exerts influence on some specific applications, users can exchange the inner pins of UART when initializing, that is to say, exchange U0TXD, U0RXD with U0RTS, U0CTS.
I2S	**I2S Input:** IO12 (I2SI_DATA) ; IO13 (I2SI_BCK); IO14 (I2SI_WS); **I2S Output:** IO15 (I2SO_BCK); IO3 (I2SO_DATA); IO2 (I2SO_WS).	I2S interface is mainly used for collecting, processing, and transmission of audio data.

5. Absolute Maximum Ratings

Table 7 Absolute Maximum Ratings

Rating	Condition	Value	Unit
Storage Temperature		-40 to 125	℃
Maximum Soldering Temperature		260	℃
Supply Voltage	IPC/JEDEC J-STD-020	+3.0 to +3.6	V

6. Recommended Operating Conditions

Table 8 Recommended Operating Conditions

Operating Condition	Symbol	Min	Typ	Max	Unit
Operating Temperature		-40	20	125	℃
Supply voltage	VDD	3.0	3.3	3.6	V

Note: Test conditions: VDD = 3.3V, Temperature = 20 ℃, if nothing special is stated.

Power Consumption

Parameters	Min	Typical	Max	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm		170		mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm		140		mA
Tx 802.11n, MCS7, P OUT =+13dBm		120		mA
Rx 802.11b, 1024 bytes packet length , -80dBm		50		mA
Rx 802.11g, 1024 bytes packet length, -70dBm		56		mA
Rx 802.11n, 1024 bytes packet length, -65dBm		56		mA
Modem-Sleep①		15		mA
Light-Sleep②		0.9		mA
Deep-Sleep③		10		uA

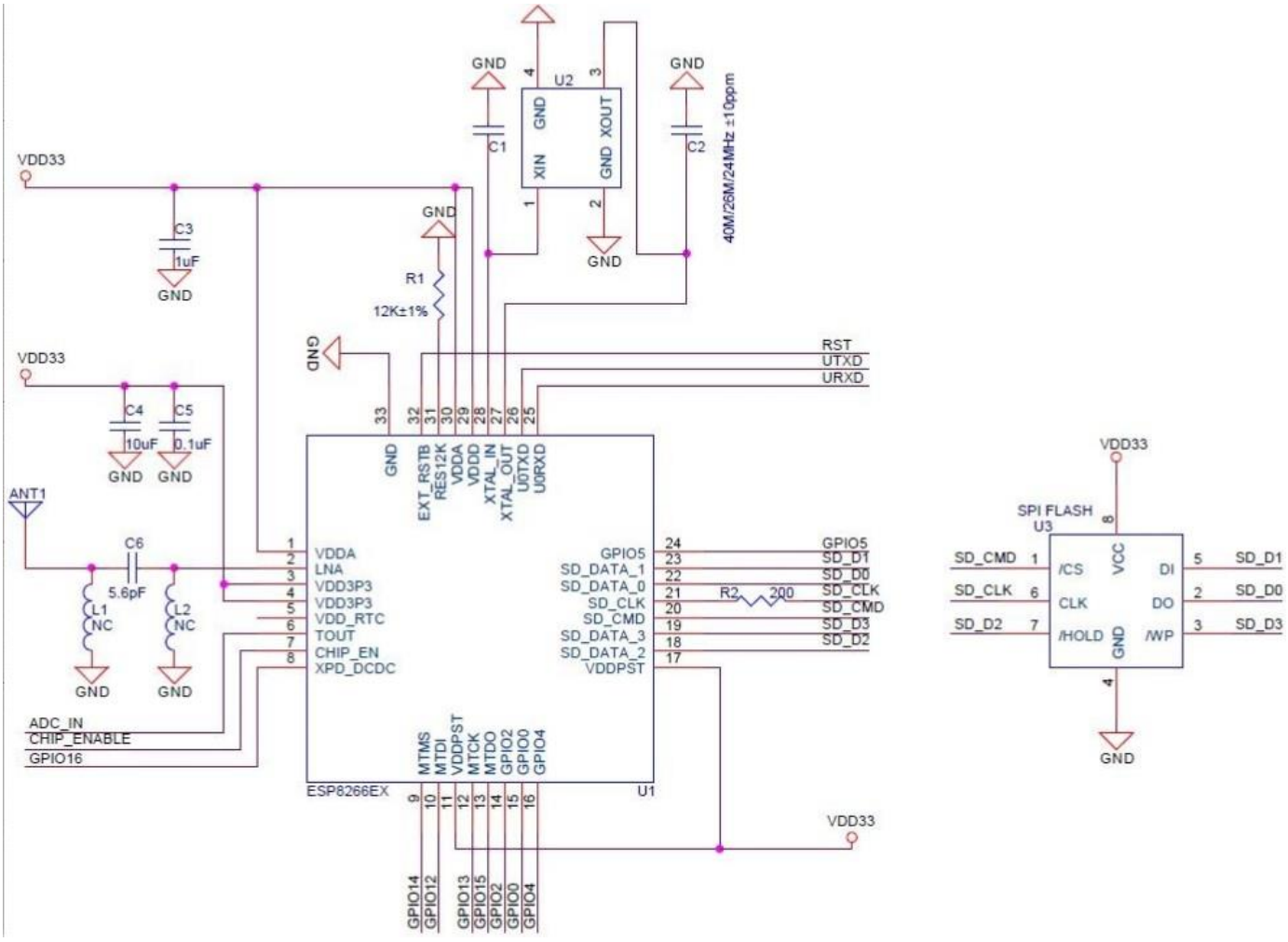
Table 11 Power Consumption

❶Modem-Sleep requires the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it saves power to shut down the W i-Fi Modem circuit while maintaining a Wi-Fi connection with no data transmission. E.g. in DTIM3, to maintain a sleep 300mswake 3ms cycle to receive AP's Beacon packages, the current is about 15mA.

❷ During Light-Sleep, the CPU may be suspended in applications like W i-Fi switch. Without data transmission, the Wi-Fi Modem circuit can be turned off and CPU suspended to save power according to the 802.11 standard (U-APSD). E.g. in DTIM3, to maintain a sleep 300ms-wake 3ms cycle to receive AP's Beacon packages, the current is about 0.9mA.

❸ Deep-Sleep does not require Wi-Fi connection to be maintained. For application with long time lags between data transmission, e.g. a temperature sensor that checks the temperature every 100s ,sleep 300s and waking up to connect to the AP (taking about 0.3~1s), the overall average current is less than 1mA.

Schematics



Relay

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protective relays".

Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect. Magnetic latching relays are useful in applications where interrupted power should not be able to transition the contacts.

Magnetic latching relays can have either single or dual coils. On a single coil device, the relay will operate in one direction when power is applied with one polarity, and will reset when the polarity is reversed. On a dual coil device, when polarized voltage is applied to the reset coil the contacts will transition. AC controlled magnetic latch relays have single coils that employ steering diodes to differentiate between operate and reset commands.

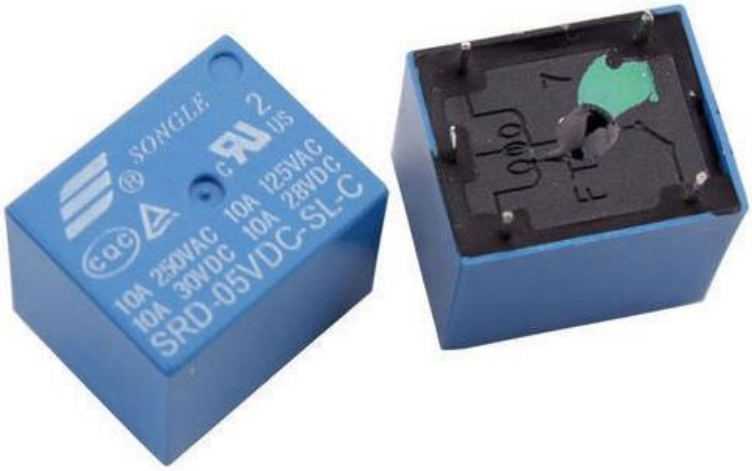


fig. 5V Relay

BT136

General description

Planar passivated sensitive gate four quadrant triac in a SOT78 plastic package intended for use in general purpose bidirectional switching and phase control applications. This sensitive gate "series E" triac is intended to be interfaced directly to microcontrollers, logic integrated circuits and other low power gate trigger circuits.

Features and benefits

- Direct triggering from low power drivers and logic ICs
- High blocking voltage capability
- Low holding current for low current loads and lowest EMI at commutation
- Planar passivated for voltage ruggedness and reliability
- Sensitive gate
- Triggering in all four quadrants

Applications

- General purpose motor control
- General purpose switching

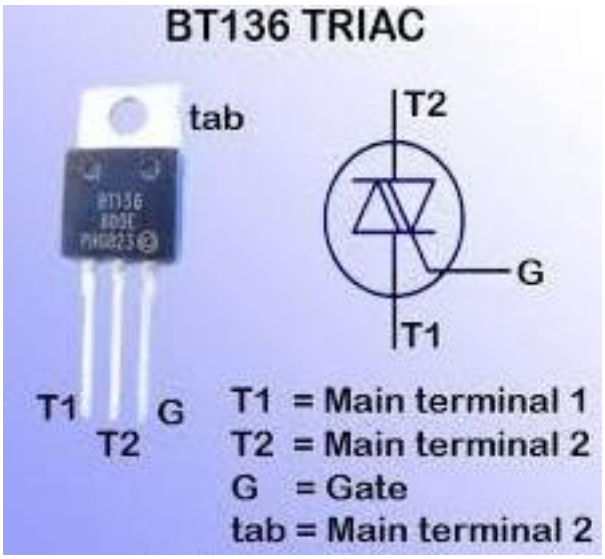


fig. BT136

MOC3020M

Description

The MOC301XM and MOC302XM series are optically isolated triac driver devices. These devices contain a GaAs infrared emitting diode and a light activated silicon bilateral switch, which functions like a triac. They are designed for interfacing between electronic controls and power triacs to control resistive and inductive loads for 115 VAC operations.

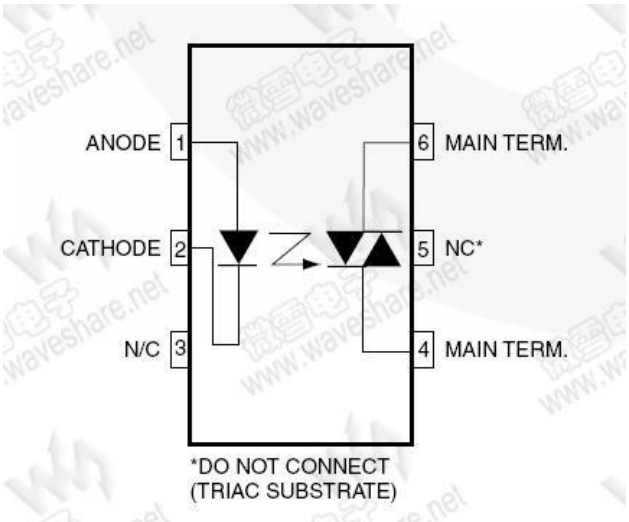
Features

- Excellent I_{FT} Stability – IR Emitting Diode has low degradation
- Peak blocking voltage
    - 250V, MOC301XM
    - 400V, MOC302XM
- Safety and regulatory approvals
    - UL1577, 4, 170 VAC_{RMS} for 1 minute
    - DIN EN/IEC60747-5-5

Applications

- Industrial Controls
- Solenoid/Valve controls
- Traffic lights
- Static AC power switch
- Vending machines
- Incandescent lamp dimmers
- Solid state relay
- Motor control
- Lamp Ballasts

Schematic



Actual figure

MCT2E

Description

Standard Single Channel Phototransistor Couplers. The MCT2/ MCTE family is an Industry Standard Single Channel Phototransistor . Each optocoupler consists of gallium arsenide infrared LED and a silicon NPN phototransistor. These couplers are Underwriters Laboratories (UL) listed to comply with a 5300 VRMS isolation test voltage. This isolation performance is accomplished through Vishay double molding isolation manufacturing process. Compliance to DIN EN 60747-5-2(VDE0884)/ DIN EN 60747-5-5 pending partial discharge isolation specification is available for these families by ordering option 1. These isolation processes and the Vishay ISO9001 quality program results in the highest isolation performance available for a commercial plastic phototransistor optocoupler. The devices are available in lead formed configuration suitable for surface mounting and are available either on tape and reel, or in standard tube shipping containers.

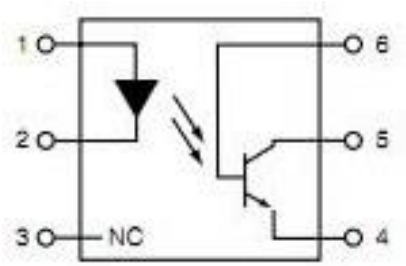
Features

- Interfaces with common logic families
- Input-output coupling capacitance < 0.5 pF
- Industry Standard Dual-in line 6-pin package
- 5300 VRMS isolation test voltage

Applications

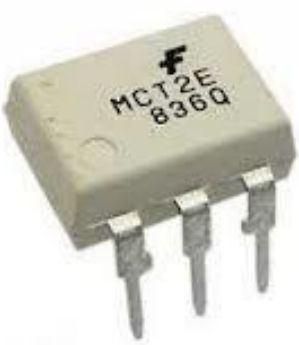
- AC mains detection
- Reed relay driving
- Switch mode power supply feedback
- Telephone ring detection
- Logic ground isolation
- Logic coupling with high frequency noise rejection

Schematic



PIN 1. ANODE
2. CATHODE
3. NO CONNECTION
4. EMITTER
5. COLLECTOR
6. BASE

Actual figure



PIC12F675

PIC12F675 is a 8 pin flash based 8 bit CMOS microcontroller which have 1024 flash words (program memory), 64 bytes SRAM &128 bytes EEPROM which are data memory. It have 8 i/o pins, 4 channel 10 bit A/D converter, 1 comparator and 1/1 8/16 bit timers.

High-Performance RISC CPU:

• Only 35 Instructions to Learn - All single-cycle instructions except branches

• Operating Speed:

- DC – 20 MHz oscillator/clock input
- DC – 200 ns instruction cycle

• Interrupt Capability

• 8-Level Deep Hardware Stack

• Direct, Indirect, and Relative Addressing modes Special Microcontroller

Features:

• Internal and External Oscillator Options

- Precision Internal 4 MHz oscillator factory calibrated to ±1%
- External Oscillator support for crystals and resonators
- 5 μs wake-up from Sleep, 3.0V, typical

• Power-Saving Sleep mode

• Wide Operating Voltage Range – 2.0V to 5.5V

• Industrial and Extended Temperature Range

• Low-Power Power-on Reset (POR)

• Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)

• Brown-out Detect (BOD)

• Watchdog Timer (WDT) with Independent Oscillator for Reliable Operation

• Multiplexed MCLR/Input Pin

• Interrupt-on-Pin Change

• Individual Programmable Weak Pull-ups

• Programmable Code Protection

• High Endurance Flash/EEPROM Cell

- 100,000 write Flash endurance
- 1,000,000 write EEPROM endurance
- Flash/Data EEPROM Retention: > 40 years

Low-Power Features:

• Standby Current: - 1 nA @ 2.0V, typical

• Operating Current:

- 8.5 μA @ 32 kHz, 2.0V, typical
- 100 μA @ 1 MHz, 2.0V, typical

• Watchdog Timer Current - 300 nA @ 2.0V, typical

• Timer1 Oscillator Current: - 4 μA @ 32 kHz, 2.0V, typical

Peripheral Features:

• 6 I/O Pins with Individual Direction Control

• High Current Sink/Source for Direct LED Drive

• Analog Comparator module with:

- One analog comparator
- Programmable on-chip comparator voltage reference (CVREF) module
- Programmable input multiplexing from device inputs
- Comparator output is externally accessible

• Analog-to-Digital Converter module (PIC12F675):

- 10-bit resolution
- Programmable 4-channel input
- Voltage reference input

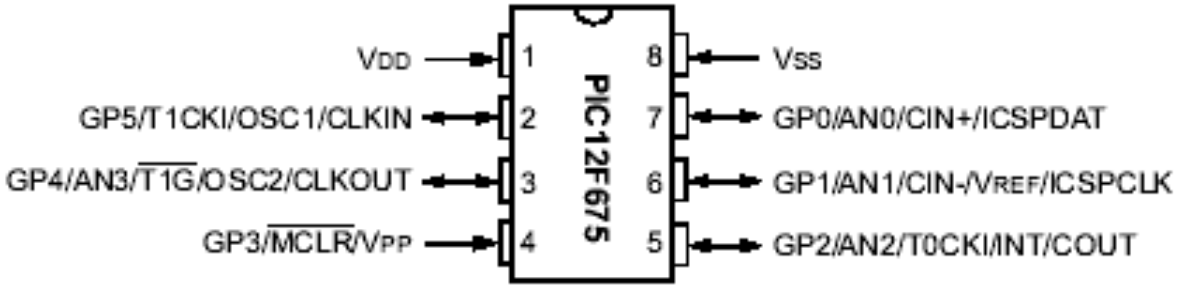
• Timer0: 8-Bit Timer/Counter with 8-Bit Programmable Prescaler

• Enhanced Timer1:

- 16-bit timer/counter with prescaler
- External Gate Input mode
- Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator, if INTOSC mode selected

• In-Circuit Serial ProgrammingTM (ICSPTM) via two pins

Schematic



Actual figure

CIRCUIT DIAGRAM

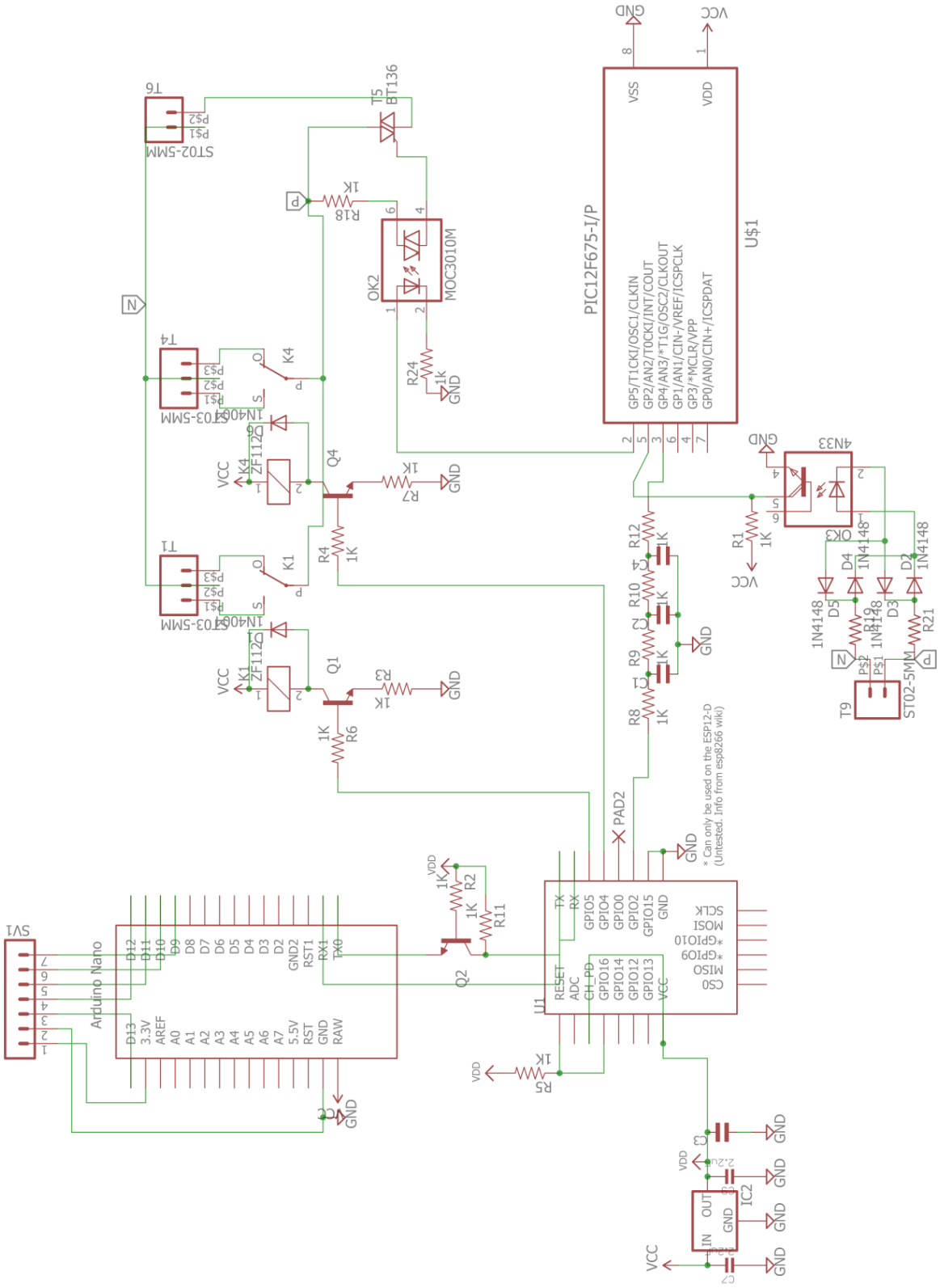


fig. circuit diagram of main module

PCB DESIGN

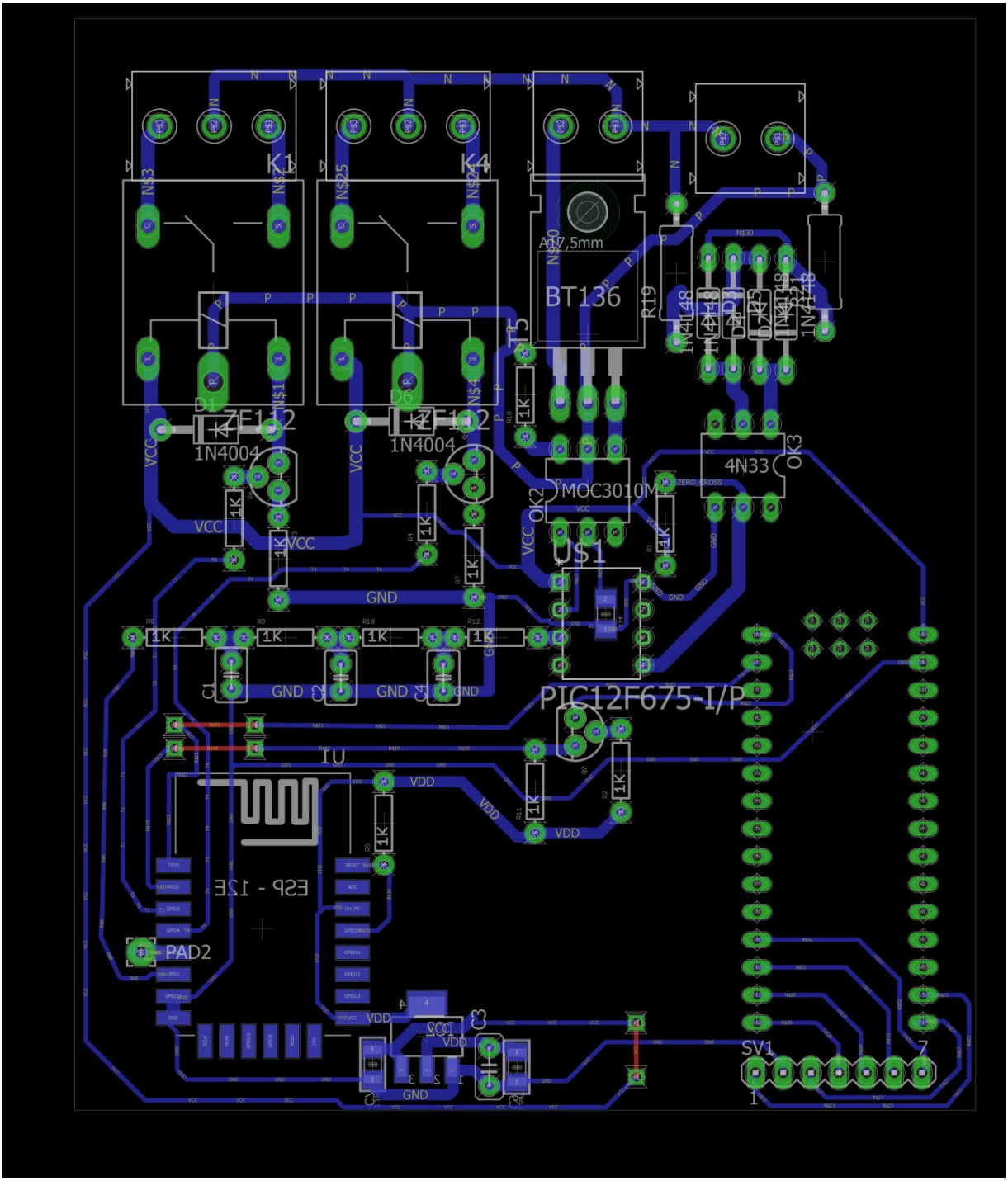


fig. pcb design of main module

CIRCUIT EXPLANATION

- 2 Channel 5V relay module

First thing is we used a 5V relay in this project. This makes it easy as Arduino can power up the relay directly. If you used a 12V relayyou need to use a separate power supply for relay.

Coming to the design of the circuit, it is very simple as we used a 5V relay module and not the individual components. Although the circuit diagram explains the detailed connections, practically we didn't need to make all the connections.

Most relay modules (whether 5V or 12V) will come with the aforementioned connection and hence all you need is to give power supply to the relay module i.e. 5V and GND and connect the control signal from Arduino to control pin on the relay board.

Coming to the load part i.e. the lamp, hot wire from the mains supply is connected to one terminal of the lamp. Other terminal of the lamp is connected to Normally Open (NO) contact of the 5V Relay. Finally, the neutral wire from the mains is connected to the Common (COMM) contact of the relay.

A small light sensor in the form of LDR (Light Dependent Resistor) is used to switch on or off the light automatically. The output of the LDR sensor is given to the Analog Input pin A0.

- Zero crossing circuit & detection

A **zero cross circuit** (or zero crossing circuit) is an electrical circuit that starts operation with the AC load voltage at close to 0 Volts in the AC cycle. This is in relation to solid state relays, such as triacs and silicon controlled rectifiers. The purpose of the circuit is to start the triac conducting very near the time point when the load voltage is crossing zero volts (at the beginning or the middle of each AC cycle represented by a sine wave), so that the output voltage begins as a complete sine-wave half-cycle. In other words, if the controlling input signal is applied at any point during the AC output wave other than very close to the zero voltage point of that wave, the output of the switching device will "wait" to switch on until the output AC wave reaches its next zero point. This is useful when sudden turn-on in the middle of a sine-wave half cycle could cause undesirable effects like high frequency spikes for which the circuit or the environment is not expected to handle gracefully.

The point where the AC line voltage is 0 V is the *Zero Cross Point*. When a triac is connected in its simplest form, it can clip the beginning of the voltage curve, due to the minimum gate voltage of the triac. A zero cross circuit works to correct this problem, so that the triac functions as well as possible. This is typically done with thyristors in two of the three phases.

Many opto-triacs come with zero cross circuits built in. They are often used to control larger, power triacs. In this setup triac turn-on delays will compound, so quick turn on times are important.

The corresponding phase angle circuits are more sophisticated and more expensive than zero cross circuits.

- Function of ESP-12E

In our project, ESP12E plays a major role, because, it is the key component by which most of the important function is done. So, if you see the block diagram above, you can easily understand that ESP12E wifi module transmit and receive data simultaneously , just like router. A router receives internet via Ethernet cable and transmit the internet service to all the direction wirelessly. Here, our ESP12E receives all the data from other 3 modules via arduino nano send it to the database via internet and the database then force the application to update those data. And when we tap anything in our corresponding application like toggling relays or, controlling brightness of a bulb then the application sends some specific data to the database and database force the ESP12E to do the respective job via internet. To this, first we have to supply internet from a device and the device's wifi hotspot should be turned on. Now our job is to connect ESP12E to the internet. The ESP12E automatically connect to our device's hotspot because the ip address of our device is hard-coded. That means ESP12E now connected to the internet. If there is many device , then the code should be dynamic , then ssid (name of the network) and password of the respective device should given to the mobile application login screen and continue to the the next page of the application. Further information are given in the software design part.

- Interfacing nRF24L01 receiver Sensor to Arduino NANO

details of the Pinout and connections :

Signal	Arduino pin for TMRh20 RF24 Library
GND	GND
VCC	3.3 V
CE	7
CSN	8
SCK	13
MOSI	11
MISO	12
IRQ	-

ESP-12E Wi-Fi module PROGRAMMING

```
#include <FirebaseArduino.h>
#include "DHT.h"
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

#define PINA 16
#define DHTPIN 14    // Data Pin of DHT 11 , for NodeMCU D5 GPIO no. is 14
#define  relay 5
#define  relay2 4
#define pwm 2

const int httpPort = 80;
const char* host = "www.aidalabindia.com";

#define FIREBASE_HOST "esptest-608f8.firebaseio.com"

String Relay;
String Relay2;
String Pwm;

String pir = "";
String tmp = "";
String gas = "";
String wat = "";

String ssid = "";
String pass = "";

const char *myssid = "ESP_HomeAutomation";
const char *password = "12345687";
ESP8266WebServer server;
IPAddress ip(192, 168, 11, 4);
IPAddress gateway(192, 168, 11, 1);
IPAddress subnet(255, 255, 255, 0);

void setup() {
  WiFi.mode(WIFI_AP_STA);
  Serial.begin(115200);
  pinMode(relay, OUTPUT);
  pinMode(relay2, OUTPUT);
  pinMode(PINA, OUTPUT);
  digitalWrite(PINA, HIGH);
  digitalWrite(relay, LOW);
  digitalWrite(relay2, LOW);
  WiFi.begin(ssid.c_str(), pass.c_str());
  Firebase.begin(FIREBASE_HOST);
  server.on("/", []() {
    server.send(200, "text/html", "<strong>HelloWorld!</strong>");
  });
  server.on("/setSSIDPASS", SSIDPASS);
  server.begin();
```

```
  WiFi.softAPConfig(ip, gateway, subnet);
  WiFi.softAP(myssid, password);
}

void loop() {
  int retry = 0;
  while (WiFi.status() != WL_CONNECTED)
  {
    retry++;
    if ((retry % 20) == 0)
      Serial.println(".");
    else
      Serial.print(".");
    delay(500);
  }

  Relay = (Firebase.getString("relay"));
  Relay2 = (Firebase.getString("relay2"));
  Pwm = (Firebase.getString("pwm").toInt());
  int rel = Relay.toInt();
  int rel2 = Relay2.toInt();
  int ppwm = Pwm.toInt();
  if (rel == 1) {
    digitalWrite(relay, HIGH);
  }
  else
  {
```

```
    digitalWrite(relay, LOW);
  }
  if (rel2 == 1) {
    digitalWrite(relay2, HIGH);
  }
  else
  {
    digitalWrite(relay2, LOW);
  }

  analogWrite(pwm, ppwm);

  Serial.print(rel);
  Serial.print(" ");
  Serial.print(rel2);
  Serial.print(" ");
  Serial.println(ppwm);
  ////////////////////////////////////////
  if (Serial.available())
  {
    String paramiters = Serial.readStringUntil('\n');
    if (paramiters.startsWith("pir"))
    {
      pir = paramiters.substring(3);
      Firebase.setString("intruder", pir);
      Serial.println("V=" + pir + "");
    }
```

```
    else if (paramiters.startsWith("tmp"))
    {
      tmp = paramiters.substring(3);
      Firebase.setString("temp", tmp);
      Serial.println("I=" + tmp + "");
    }
    else if (paramiters.startsWith("gas"))
    {
      gas = paramiters.substring(3);
      Firebase.setString("gas", gas);
      Serial.println("P=" + gas + "");
    }
    else if (paramiters.startsWith("wat"))
    {
      wat = paramiters.substring(3);
      Firebase.setString("waterl", wat);
      Serial.println("KWH=" + wat + "");
    }
    WiFiClient client;
    if (!client.connect(host, httpPort)) {
      Serial.println("connection failed");
      return;
    }
  }
  server.handleClient();
}
```

```
void SSIDPASS()
{
  setup();
  String ssid = server.arg("SSID");
  String pass = server.arg("PASS");
  WiFi.begin(ssid.c_str(), pass.c_str());
  // Use WiFi Object Begin Methode To ConnectTo AccesssPoint
  Serial.println(ssid);
  Serial.println(pass);
  char D = WiFi.localIP();
   digitalWrite(PINA, LOW);
  server.send(200, "text/html", "");
}
```

Arduino Nano Programming

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
// include nRF24L01 header file

RF24 radio(7,8); // define CE, CSN pins
const byte address[] = {0xB00B1E50D2LL, 0xB00B1E50C3LL, 0xB00B1E50B1LL};
// define specific pipelines for communicate
float gotByte;
int state ;
int sense ;

void setup() {
  Serial.begin(115200); //serial monitor at 115200 baud rate
  radio.begin(); //start communication
  radio.openReadingPipe(1, address[0]);
  radio.openReadingPipe(2, address[1]);
  radio.openReadingPipe(3, address[2]);
  //start communcating data at defined pipelines
  radio.setPALevel(RF24_PA_HIGH); //set range of the Rx
  radio.startListening(); // start receiving data
}

void loop() {
  byte pipeNum = 0;
  while (radio.available(&pipeNum)) // check if data is available or not

  {
    if (pipeNum == 1)
    {
      radio.read( &gotByte, sizeof(gotByte) );
      Serial.print("tmp");
      Serial.println(gotByte);
    }
    else if (pipeNum == 2)
    {
      radio.read(&sense, sizeof(sense));
      Serial.print("pir");
      Serial.println(sense);
    }
    else if (pipeNum == 3)
    {
      radio.read(&state, sizeof(state));
      Serial.print("gas");
      Serial.println(state);
    }
  }
  delay(700); // delay for 700ms.
}
```

ADVANTAGES

- The given system is handy and portable, and thus can be easily carried from one place to another.
- Though the circuitry is complex ,but, it is easy to understand.

DISADVANTAGES

- For making prototype project ESP12E wifi module is not so good, because, it has limited no. of ports, and power issues.
- nRF24L01 is good to transmit large amount of data, but, it has power stability issue for long range communication.

POSSIBILITIES

- As a replacement nodemcu / adafruitesp can be used. Also, esp32 ,the successor of esp8266 can be used.
- The data transmission link should be more reliable. Thus, I prefer to use more stable components and communication protocol like Xbee.

APLLICATIONS

This module ,itself has a defined work which is to control AC appliances. That work is flawlessly done and can be implemented anywhere. Overall it is actually a main module where many peripheral modules are needed. So, any work can be done by this module , the possibilities are endless.

2nd Module (Gas detection and alarming system)

• Leakage of LPG can be dangerous as it raises the risk of building fire, suffocation or an explosion.

• For every 20L volume of air, 2% to 10% of LPG in air is enough to cause an explosion.

• The Chiba Oil Refinery fire incident in japan on 11th March, 2011 which caused six injuries and destroyed all 17 LPG tanks comes to mind.

• In such a case, a gas leakage detector becomes vital and helps to protect people from the dangers of gas leakage.

• Thus, our project is aimed at designing and building a wireless LPG leakage detector.

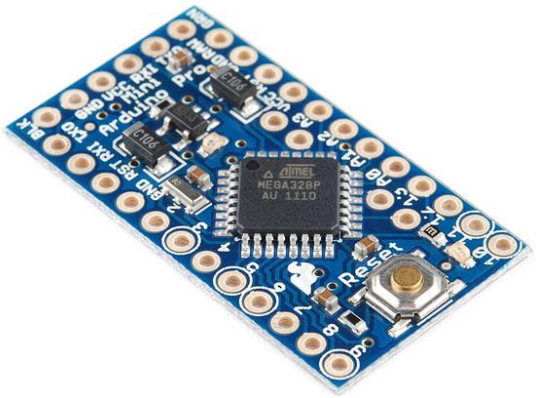
OBJECTIVE

- Detect Gas Leakage (like LPG and smoke) using MQ-6 Sensor and Arduino .
- Setup an Wi-Fi and IoT (Internet of Things) based Alert Mechanism using ESP-8266 Wi-Fi Module.
- Send alert messages to specified android mobile which are connected via Wi-Fi or Internet.
- LED indication – In, normal condition a green led is always ON, but , when the module detects gas leakage red led is turned ON and green led is OFF.
- Sound Alarm – produce sound alert on gas leak using buzzer.
- Window opener – Automatically opens window of the specific area, so that, gas or smoke can go outside.
- Exhaust Fan – Automatically start the exhaust fan.
- Display status in specified android mobile.

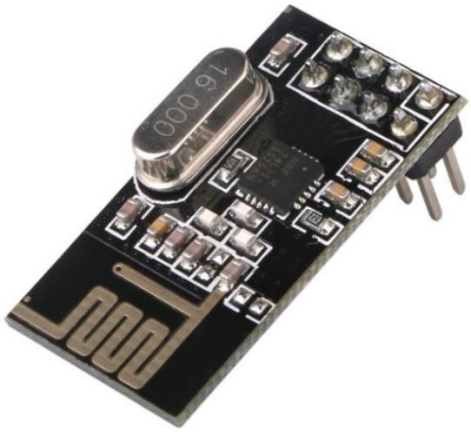
PLANNING OF COMPONENTS

- **MQ-6** – to sense LPG and smoke. (using Analog Output to detect level status)

- **Arduino Pro MINI**
  - To read MQ-6 output and detect Gas Leak (through level comparison).
  - To activate outputs upon gas leak – LED indication, sound alarm, open window and start exhaust fan.
  - To send alert Commands to Wi-Fi Module.
  - To send Status message commands to Wi-Fi Module.



- **NRF24L01 Module** – for transmitting the data received from the temperature sensor .

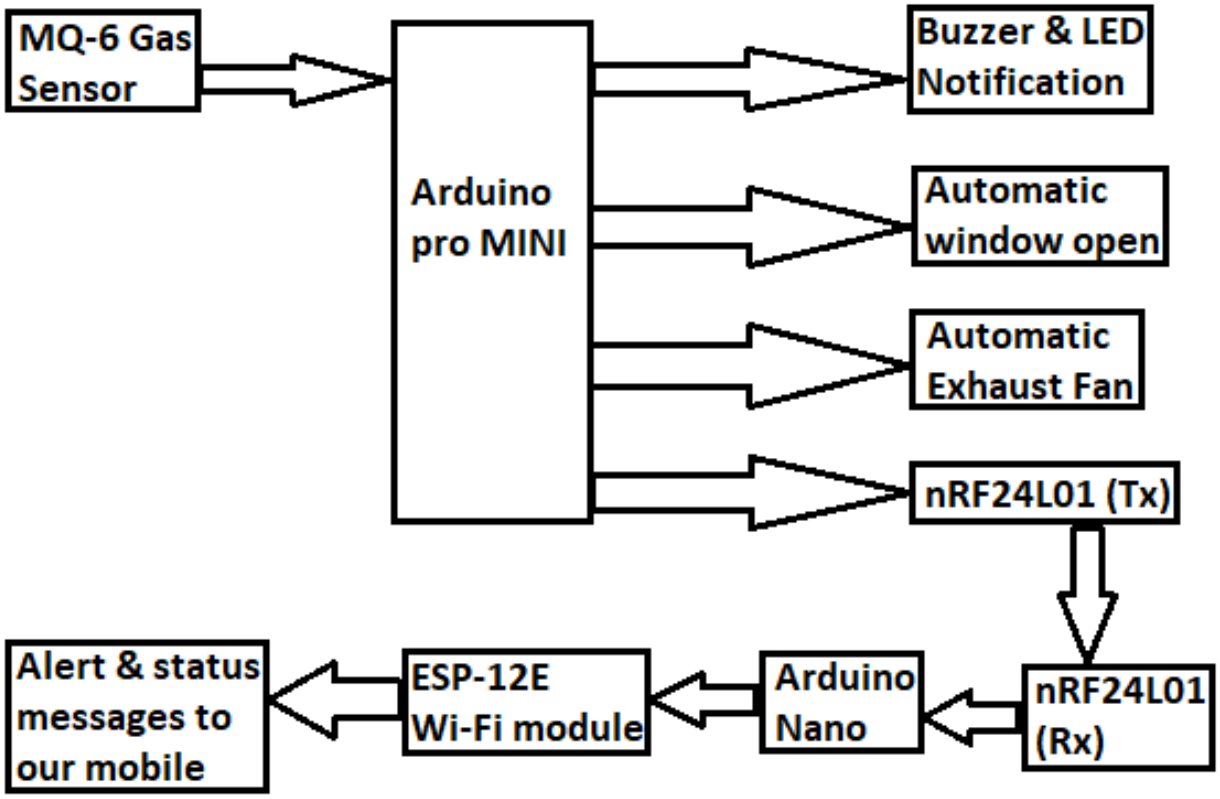


- **Window opener** – we use servo motor which is attached to the window to open and close the window.

- **Exhaust fan** – Here, we use a normal dc fan for demonstration purpose.

- **Speaker** – to produce Sound alarm using buzzer.

BLOCK DIAGRAM



COMPONENTS DESCRIPTION

MQ-6 Gas Sensor Module

The MQ series of gas sensors use a small heater inside with an electro-chemical sensor. They are sensitive for a range of gasses and are used indoors at room temperature.They can be calibrated more or less (see the section about "Load-resistor" and "Burn-in") but a know concentration of the measured gas or gasses is needed for that.

The output is an analog signal and can be read with an analog input of the Arduino.

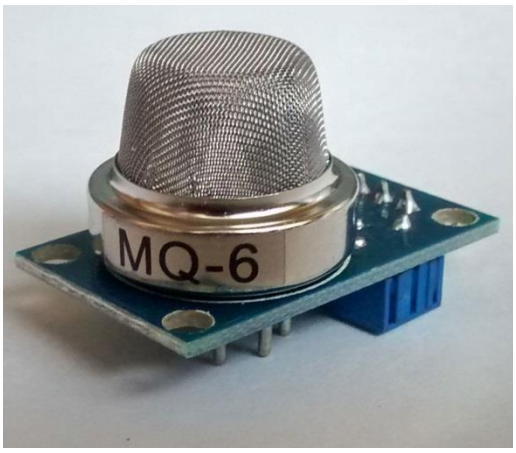


fig. MQ-6 gas sensor module

Wiring

The preferred wiring is to connect both 'A' pins together and both 'B' pins together. It is safer and it is assumed that is has more reliable output results. Although many schematics and datasheets show otherwise, you are advised to connect both 'A' pins together and connect both 'B' pins together.

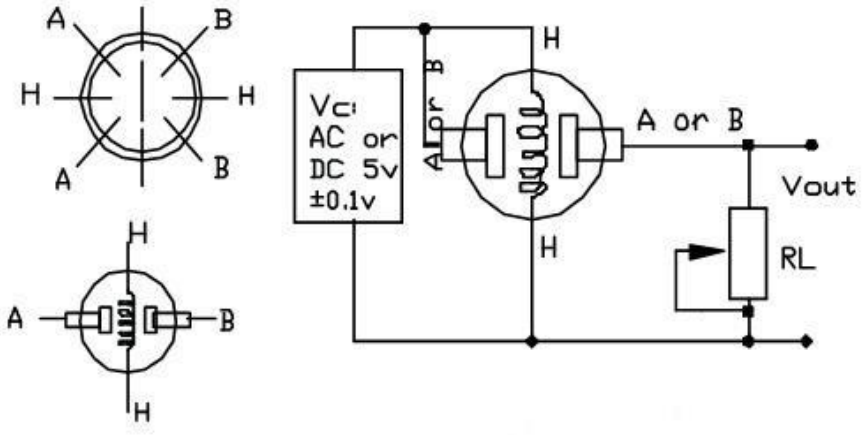


fig. internal components & connections of a gas sensor

In the picture, the heater is for +5V and is connected to both 'A' pins. This is only possible if the heater needs a fixed +5V voltage.

The variable resistor in the picture is the load-resistor and it can be used to determine a good value. A fixed resistor for the load-resistor is used in most cases.

The Vout is connected to an analog input of the Arduino.

The heater

The voltage for the internal heater is very important. Some sensors use 5V for the heater, others need 2V. The 2V can be created with a PWM signal, using analogWrite() and a transistor or logic-level mosfet. The heater may not be connected directly to an output-pin of the Arduino, since it uses too much current for that.

Some sensors need a few steps for the heater. This can be programmed with an analogWrite() function and delays. A transistor or logic-level mosfet should also in this situation be used for the heater.

If it is used in a battery operated device, a transistor or logic-level mosfet could also be used to switch the heater on and off.The sensors that use 5V or 6V for the internal heater do get warm. They can easily get 50 or 60 degrees Celcius.

After the "burn-in time", the heater needs to be on for about 3 minutes (tested with MQ-2) before the readings become stable.

Load-resistor

The sensor needs a load-resistor at the output to ground. It's value could be from 2kOhm to 47kOhm. The lower the value, the less sensitive. The higher the value, the less accurate for higher concentrations of gas.

If only one specific gas is measured, the load-resistor can be calibrated by applying a know concentration of that gas. If the sensor is used to measure any gas (like in a air quality detector) the load-resistor could be set for a value of about 1V output with clean air.

Choosing a good value for the load-resistor is only valid after the burn-in time.

Burn-in

Some datasheets use the term "preheat", but it is the time to burn-in the sensor. This is meant to make the sensor readings more consistent. A time of 12 or 24 hours is usually used for the burn-in time.

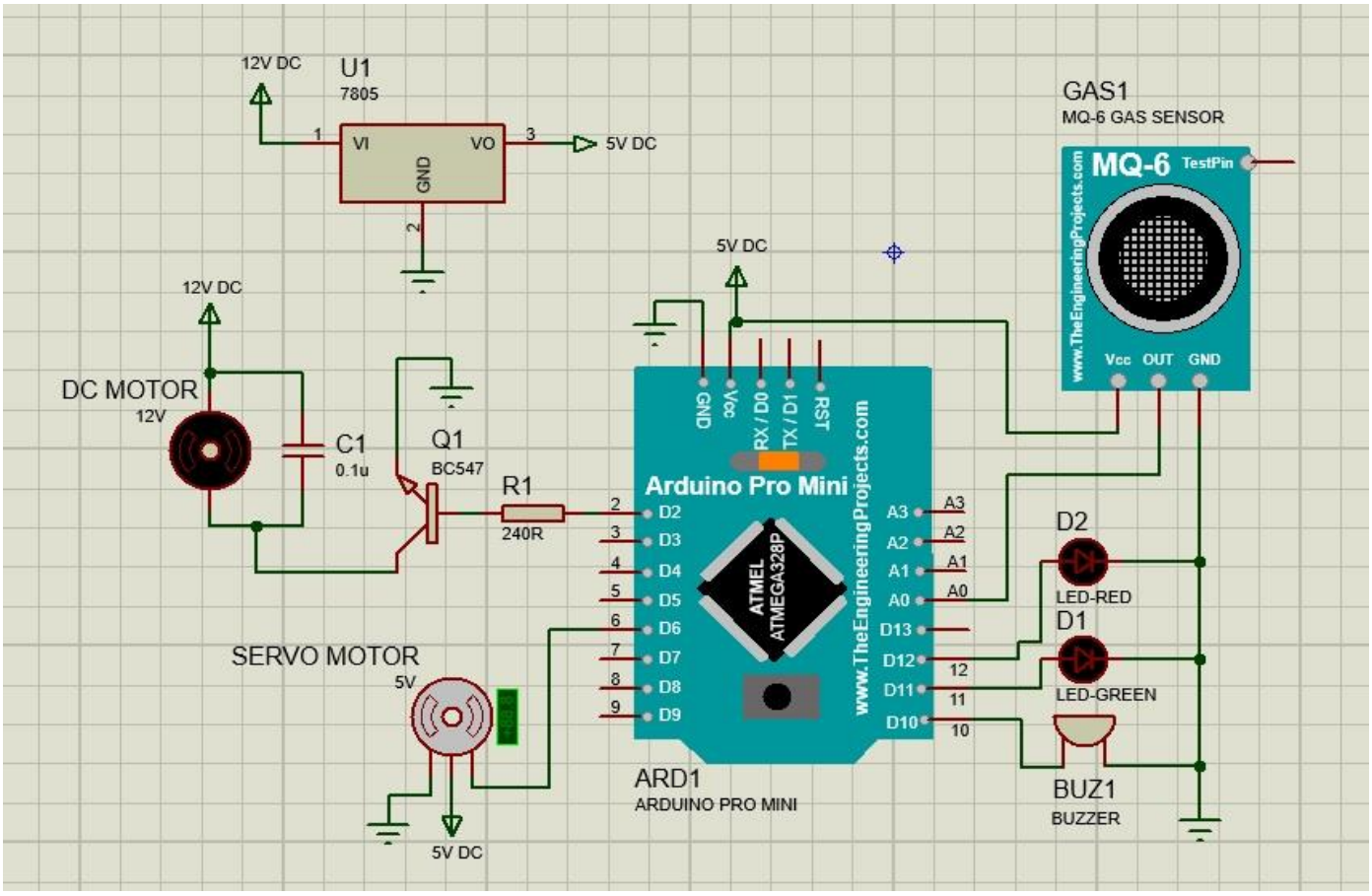
The Burn-in is achieved by applying normal power to the sensor (to the heater and with the 'A' and 'B' pins connected, and with a load-resistor). In some special cases a specific burn-in is needed. See the datasheet if the sensor needs such a specific burn-in.

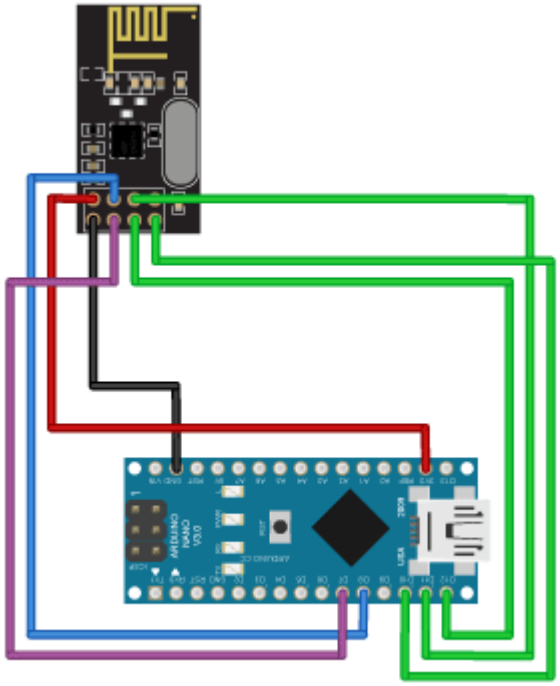
MQ-6

Sensitive for LPG, butane gas, The heater uses 5V.

The MQ-6 at seeed: http://www.seeedstudio.com/wiki/Electronic_brick_-_Gas_sensor%28MQ6%29
The MQ306A (also on this page) is like this sensor, but uses a lower heater voltage.earch for datasheet: http://duckduckgo.com/?q=%22mq6%22+gas+sensor+filetype%3Apdf

CIRCUIT DIAGRAM





CIRCUIT EXPLANATION

• Interfacing MQ-6 Sensor to Arduino

- Connecting Analog Out of MQ-6 to pin A0

• Interfacing 12V DC motor (which is used as an exhaust fan) to Arduino

- Connect 240 ohm resistance to digital pin 2 of arduino and the other end of the resistor goes to the base of a transistor BC547B.
- Here, BC547B transistor used as a amplifier to drive the 12V DC motor. Collector pin is connected to one of the end of DC motor.
- 0.1uF capacitor is connected across the 12V DC motor.
- The other end of 12V DC motor is directly connected to the power supply.

• Interfacing a 5V servo motor (which is used as a window opener) with arduino – control pin of servo motor is connected to PWM pin 6 of the arduino.

• Interfacing buzzer to digital pin 10 of Arduino.

• Interfacing indication LED to digital pin 11 & 12 of the Arduino.

• Interfacing nRF24L01 transmitter Sensor to Arduino pro MINI

details of the Pinout and connections :

Signal	**Arduino pin for TMRh20 RF24 Library**
GND	GND
VCC	3.3 V
CE	7
CSN	8
SCK	13
MOSI	11
MISO	12
IRQ	-

ARDUINO PRO MINI PROGRAMMING

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
// include nRF24L01 header file
#include <Servo.h> // include servo motor header file

int val = 1;
Servo servo_1; // initialize servo motor
int redLed = 5; // choose the pin for the red LED
int greenLed = 2; // choose the pin for the green LED
int buzzer = 9; // choose the pin for the buzzer
int smokeD0 = 6; // choose the input pin (for MQ-6 GAS sensor)
int sensorThres = 1; // iniialize default value of gas sensor
int motor = 10; // choose the pin for the exhaust fan
RF24 radio(7, 8); // define CE, CSN pins
const byte address = 0xB00B1E50B1LL;
// define specific pipeline for communicate

void setup() {
  pinMode(smokeD0, INPUT); //declare gas sensor as i/p
  Serial.begin(115200); //serial monitor at 115200 baud rate
  pinMode(redLed, OUTPUT); //declare redLED as o/p
  pinMode(greenLed, OUTPUT); //declare greenLED as o/p
  pinMode(buzzer, OUTPUT); // declare buzzer as o/p
  pinMode(motor, OUTPUT); // declare exhaust fan as o/p
  pinMode(servo_1.attach(3), OUTPUT); //declare exhaust fan as o/p

  radio.begin(); //start communication
  radio.openWritingPipe(address);
  //start communcating data at defined pipeline
  radio.setPALevel(RF24_PA_HIGH); //set range of the Tx
  radio.stopListening(); // start transmitting data
}

void loop() {
  int digitalSensor = digitalRead(smokeD0);
  //read i/p value of gas sensor
  if (digitalSensor != sensorThres)
  { digitalWrite(redLed, HIGH); //turn redLED on
    digitalWrite(greenLed, LOW); // turn greenLED off
    digitalWrite(motor, HIGH); // turn exaust fan on
    servo_1.write(5); // turn servo motor on
    tone(buzzer, 2940, 2000);
    // turn on the buzzer at 2940Hz & for 2000 ms.
    radio.write(&digitalSensor, sizeof(digitalSensor));
    Serial.println("Gas leaked");
    delay(4000); // digital sensor is high for 4 sec.
    digitalSensor =1;
    radio.write(&digitalSensor, sizeof(digitalSensor));
    delay(500);    // delay 500 ms.
  }
  else if (digitalSensor == sensorThres)
  { digitalWrite(redLed, LOW); // turn redLED off
    digitalWrite(greenLed, HIGH); //turn greenLED on
```

```
    digitalWrite(motor, LOW); // exhaust fan is off
    servo_1.write(80); // servo motor is off
    noTone(buzzer); // turn off buzzer
  }
}
```

ADVANTAGES

- The given system is handy and portable, and thus can be easily carried from one place to another.
- The circuitry is not that complicated and thus can be easily troubleshooted.
- extra components not needed.
- In, MQ-6 gas sensor, both analog and digital signal output is provided.

DISADVANTAGES

- nRF24L01 is good to transmit large amount of data, but, it has power stability issue for long range communication.
- High power consumption due to sensor heater.
- Sensitivity range is very low.

POSSIBILITIES

- The data transmission link should be more reliable. Thus, I prefer to use more stable components and communication protocol like Xbee.
- More sensitive gas module is needed for real time implementation.

APPLICATIONS

• **Home Security –** can be used in homes (especially kitchen area) to prevent accidents due to gas leak

• **Industrial Security –** can be used in sensitive areas to prevent any accidents

• **Enhancement –** can be enhanced to measure specific gas levels to use in industrial applications

• **Automation –** can be enhanced to automate electrical cut off process to prevent short circuit.

3rd Module (Motion detection and alarming system)

The need for home security alarm systems nowadays is a serious demand. As the number of crimes are increasing every day, there has to be something that will keep us safe. We are all aware of the high end security systems present in the market but they are not easily available to everyone. We therefore intend to provide a solution by constructing a cost efficient electronic system that has the capability of sensing the motion of the intruders and setting off a notification system to you phone.

Besides, power consumption is a big issue in today's world, because, fossil fuel is limited, ending very soon and renewable energies are not enough to replace fossil fuel energies. So, to prevent this we can implement some technologies from the basic level. Using PIR sensor which has the capability to sense the motion we can control our home appliances as per our need. When someone enters in the room appliances will automatically turned on and when everyone leaves appliances will automatically turned off.

The basic idea behind this project is that all the bodies generate some heat energy in the form of infrared which is invisible to human eyes. But, it can be detected by electronic motion sensor.

The project involves the use of Arduino, PIR motion sensor, nRF24L01 Transreceiver and a simple program. The sensor detect any motion in its permissible range and send the signal to Arduino and ESP12E which processes the signal and security notification to our mobile. With this system we can easily set up a security alarm in our home for unwanted intruders.

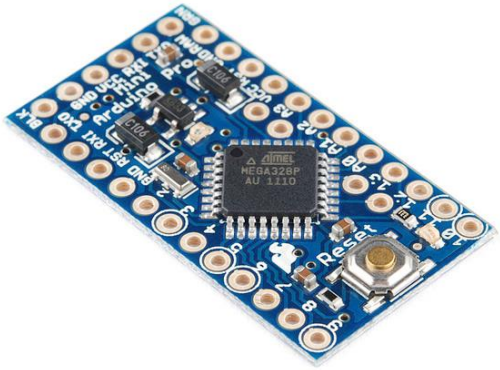
OBJECTIVE

- Detect motion using PIR Sensor (Passive Infrared Sensor) and Arduino .
- Setup an Wi-Fi and IoT (Internet of Things) based Alert Mechanism using ESP-12E Wi-Fi Module.
- Send security alert messages to specified android mobile which are connected via Wi-Fi or Internet.
- Turning on the LED – In, normal condition when there is no movement, led will turned off but, when there is an activity led will turned on for a certain time period and we can control the time period.
- Display status in specified android mobile.

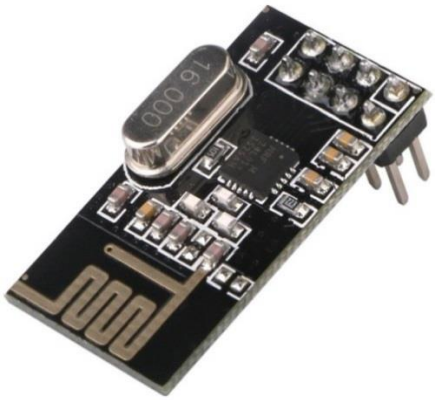
PLANNING OF COMPONENTS

- **HC-SR501 PIR Sensor** – to sense motion using infrared.

- **Arduino Pro MINI**
  - To read PIR sensor output and detect motion (through level comparison).
  - To activate outputs upon when detecting motion – turning on 5V LED.
  - To send security alert Commands to Wi-Fi Module.
  - To send Status message commands to Wi-Fi Module.

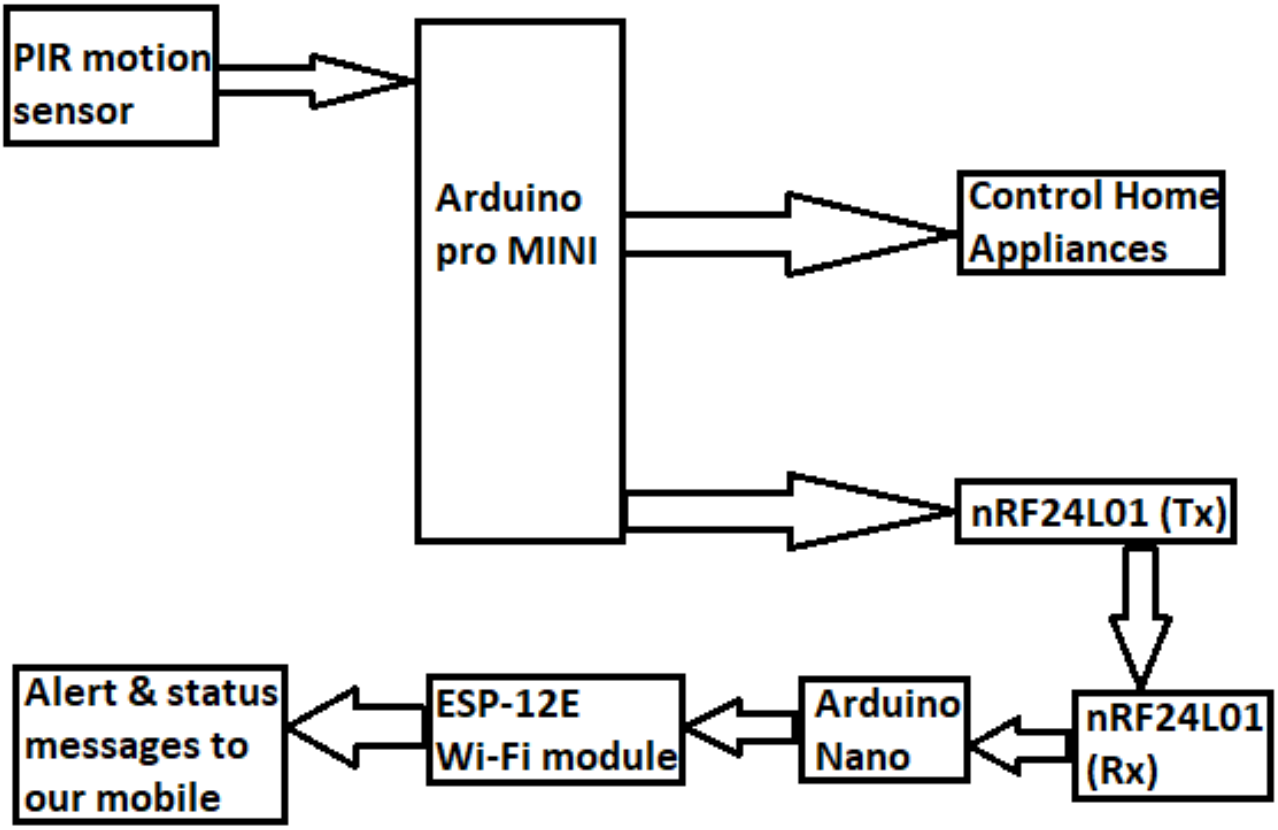


- **NRF24L01 Module** – for transmitting the data received from the PIR sensor



- **LED** – act as a alert

BLOCK DIAGRAM



COMPONENTS DESCRIPTION

HC-SR501 PIR motion sensor

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

fig. PIR sensor

PIRs are basically made of a pyroelectric sensor (which you can see below as the round metal can with a rectangular crystal in the center), which can detect levels of infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is actually split in two halves. The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low.

fig. Infrared diode

Along with the pyroelectic sensor is a bunch of supporting circuitry, resistors and capacitors. It seems that most small hobbyist sensors use the BISS0001 ("Micro Power PIR Motion Detector IC"), undoubtedly a very inexpensive chip. This chip takes the output of the sensor and does some minor processing on it to emit a digital output pulse from the analog sensor.

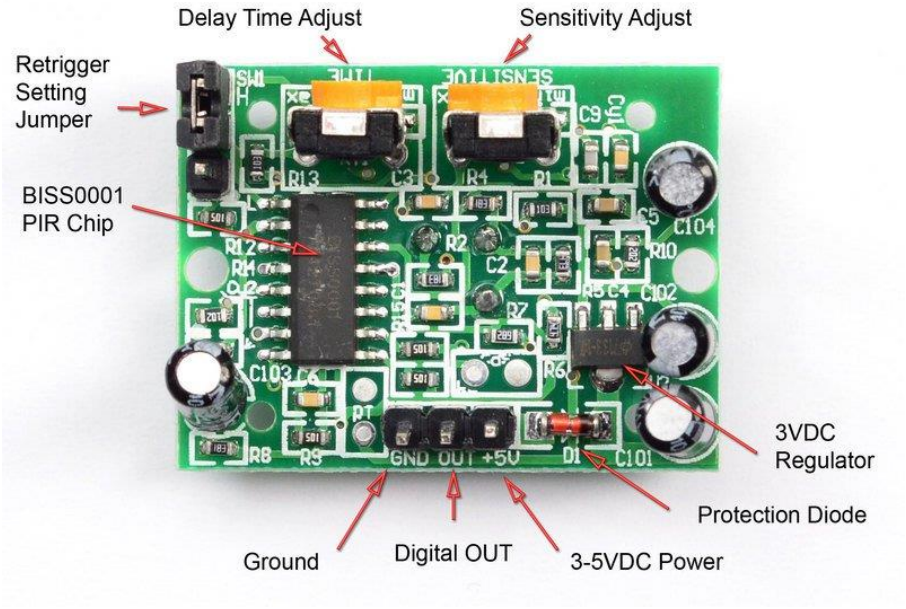


fig. Backside of PIR sensor

For many basic projects or products that need to detect when a person has left or entered the area, or has approached, PIR sensors are great. They are low power and low cost, pretty rugged, have a wide lens range, and are easy to interface with. Note that PIRs won't tell you how many people are

around or how close they are to the sensor, the lens is often fixed to a certain sweep and distance (although it can be hacked somewhere) and they are also sometimes set off by house-pets.

How PIRs Work

PIR sensors are more complicated than many of the other sensors explained in these tutorials (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output. To begin explaining how a basic sensor works, we'll use this rather nice diagram.

The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used heis not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity othe sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the
room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensinarea, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

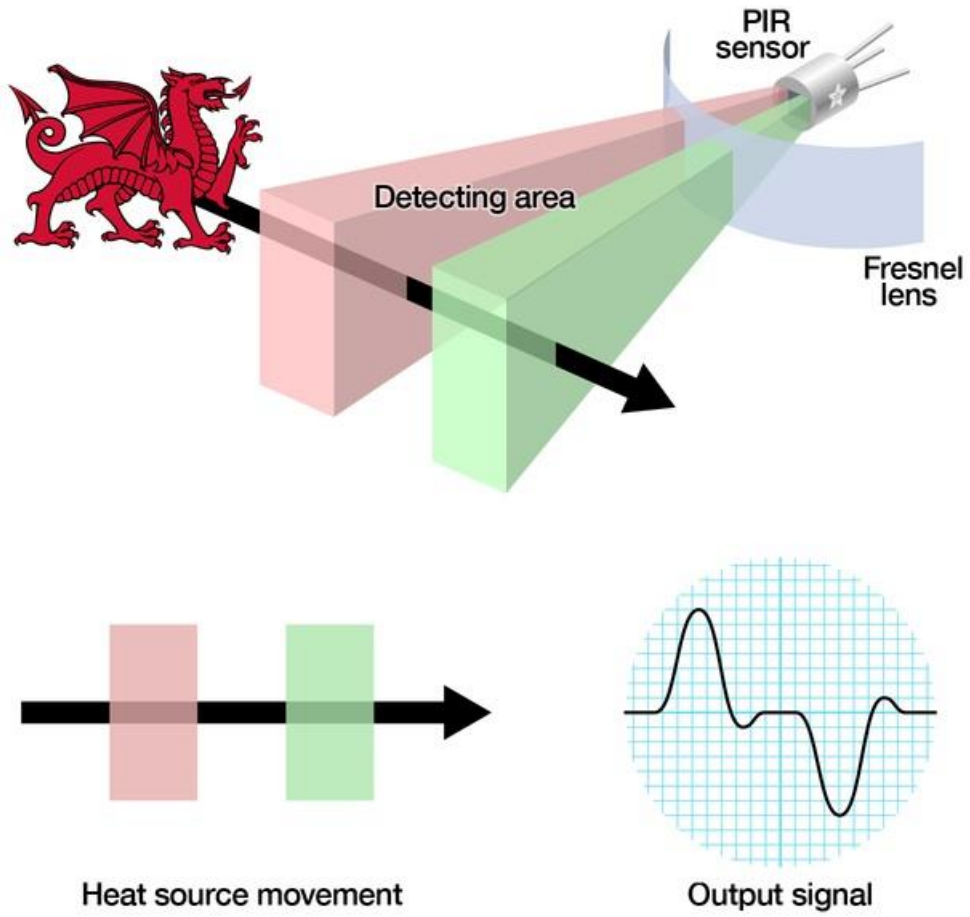


fig. How PIR works

The PIR Sensor

The IR sensor itself is housed in a hermetically sealed metal can to improve noise/temperature/humidity immunity. There is a window made of IR-transmissive material (typically coated silicon since that is very easy to come by) that protects the sensing element. Behind the window are the two balanced sensors.

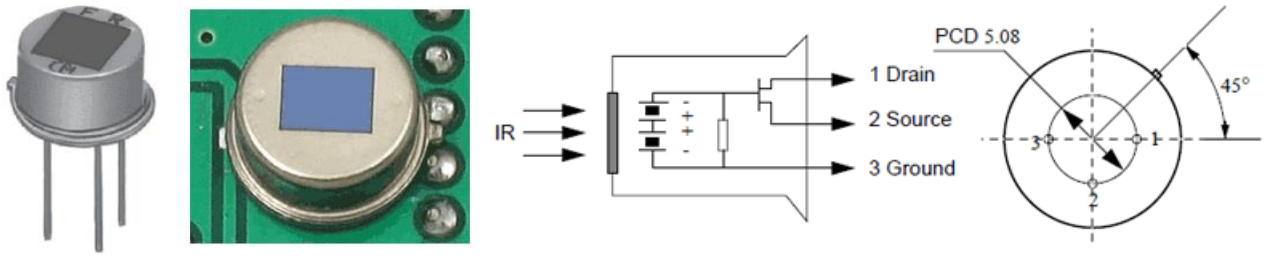
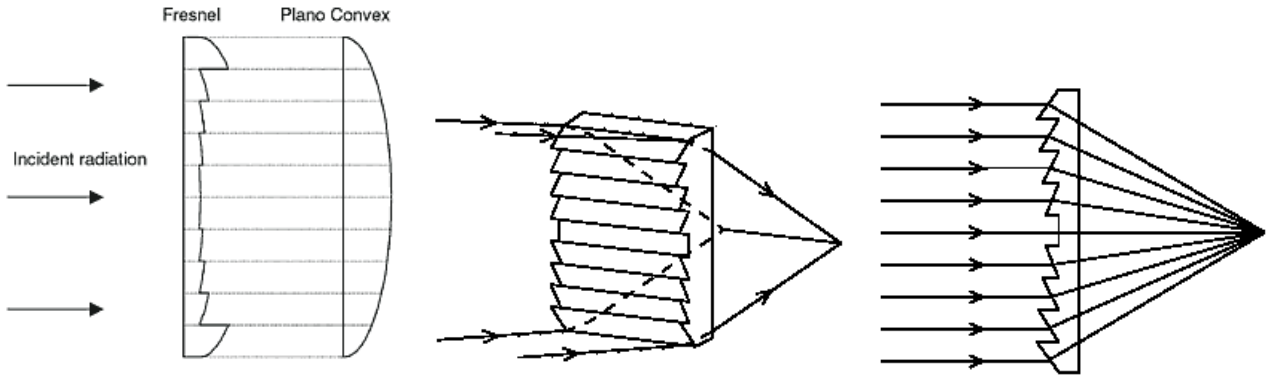


fig. The PIR sensor

This image shows the internal schematic. There is actually a JFET inside (a type of transistor) which is very low-noise and buffers the extremely high impedence of the sensors into something a low-cost chip (like the BIS0001) can sense.

Lenses

PIR sensors are rather generic and for the most part vary only in price and sensitivity. Most of the real magic happens with the optics. This is a pretty good idea for manufacturing: the PIR sensor and circuitry is fixed and costs a few dollars. The lens costs only a few cents and can change the breadth, range, sensing pattern, very easily.In the diagram up top, the lens is just a piece of plastic, but that means that the detection area is just two rectangles. Usually we'd like to have a detection area that is much larger. To do that, we use a simple lens such as those found in a camera: they condenses a large area (such as a landscape) into a small one (on film or a CCD sensor). For reasons that will be apparent soon, we would like to make the PIR lenses small and thin and moldable from cheap plastic, even though it may add distortion. For this reason the sensors are actually Fresnel lenses. The Fresnel lens condenses light, providing a larger range of IR to the sensor.



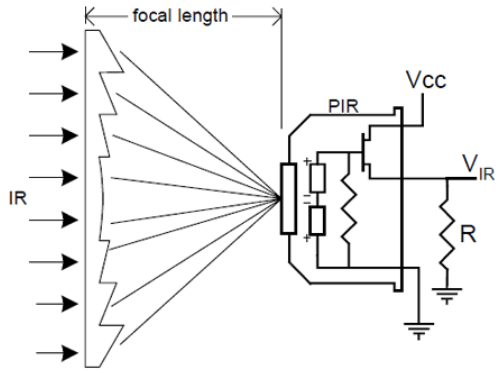


fig. Fresnel Lense

OK, so now we have a much larger range. However, remember that we actually have two sensors, and more importantly we dont want two really big sensing-area rectangles, but rather a scattering of multiple small areas. So what we do is split up the lens into multiple section, each section of which is a fresnel lens.

fig. Here you can see the multiple facet-sections

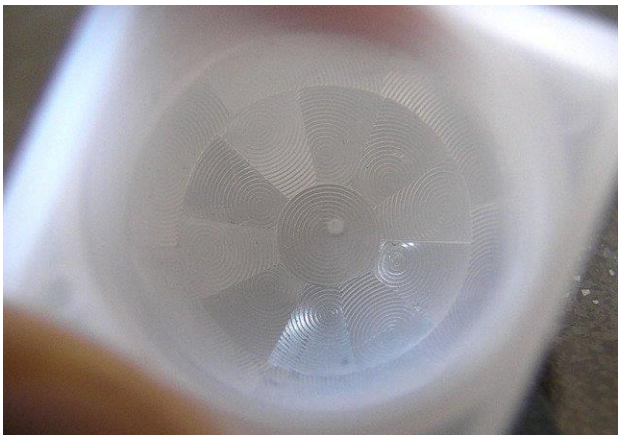


fig. This macro shot shows the different Frenel lenses in each facet!

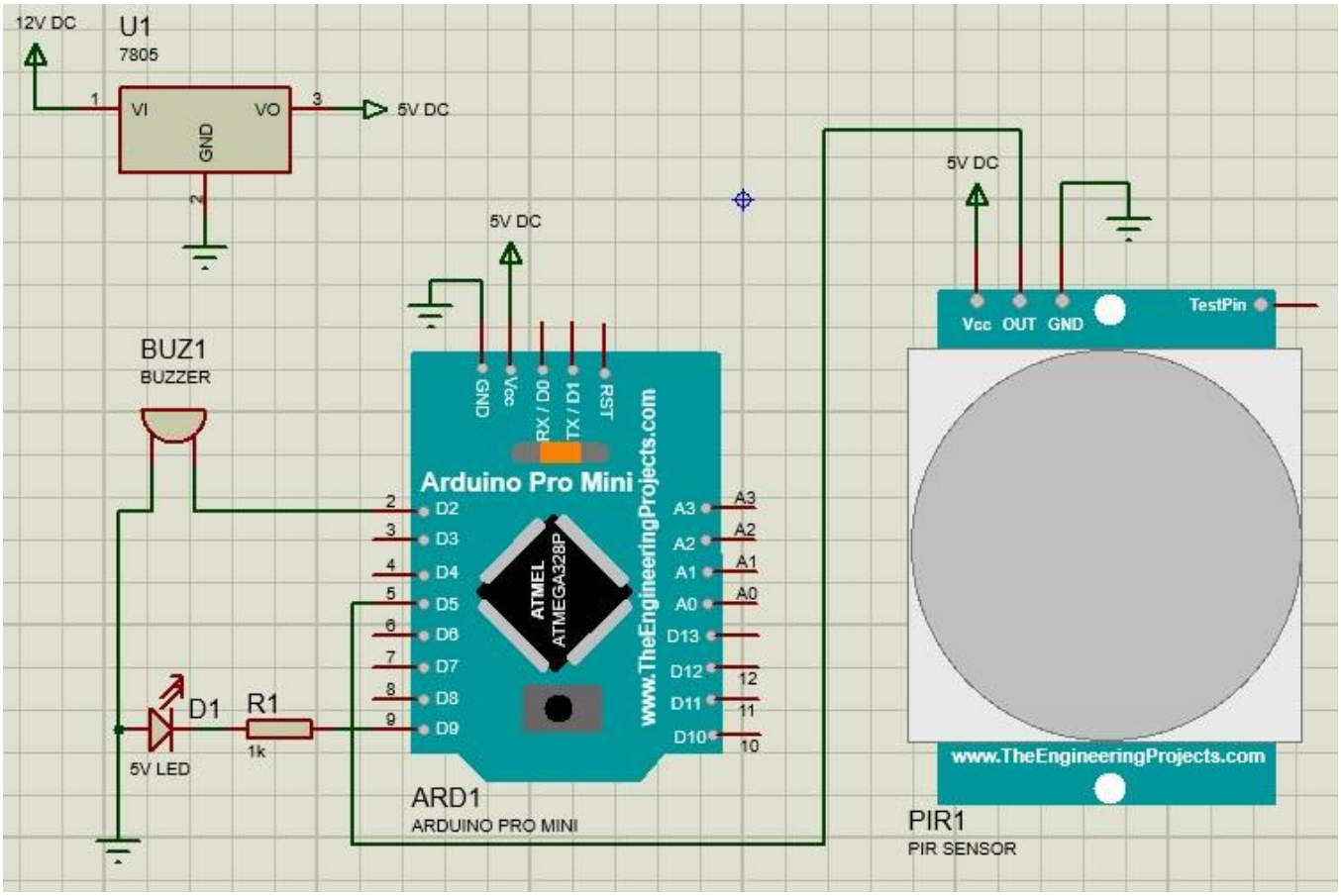
The different faceting and sub-lenses create a range of detection areas, interleaved with each other. Thats why the lens centers in the facets above are 'inconsistant' - every other one points to a different half of the PIR sensing element.

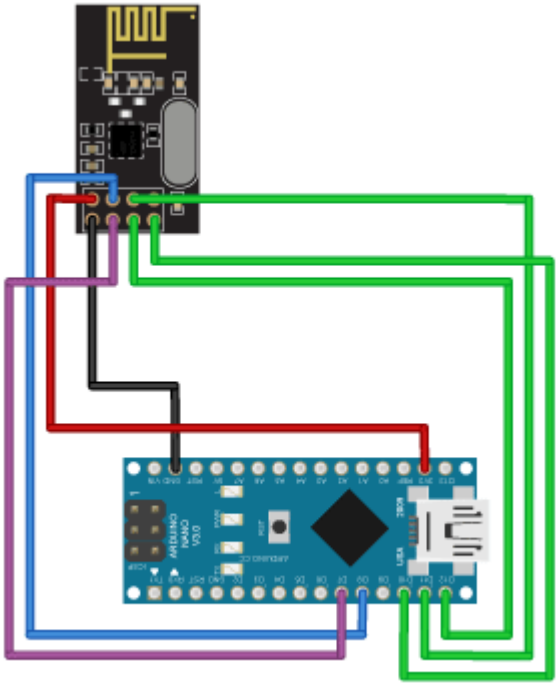
Connecting to a PIR

Most PIR modules have a 3-pin connection at the side or bottom. The pinout may vary between modules so triplecheck the pinout! It's often silkscreened on right next to the connection (at least, ours is!) One pin will be ground, another will be signal and the final one will be power. Power is usually 3-5VDC input but may be as high as 12V. Sometimes larger modules don't have direct output and instead just operate a relay in which case there is ground, power and the two switch connections.

The output of some relays may be 'open collector' - that means it requires a pullup resistor. If you're not getting a variable output be sure to try attaching a 10K pullup between the signal and power pins.

CIRCUIT DIAGRAM





CIRCUIT EXPLANATION

• Interfacing PIR Sensor to Arduino

- Connecting middle pin of HC-SR501 PIR to digital pin 9 of arduino

• Interfacing 12V LED (which is used a home appliance) to Arduino

- Connect 1.8 kilo-ohm resistance to digital pin 5 of arduino and the other end of the resistor goes to the base of a transistor TIP122.
- Here, TIP122 transistor used as a driver to drive the 12V LED. Collector pin is connected to negative end of the LED and the emitter pin connected to the ground.
- The positive end of 12V LED is directly connected to the power supply.

• Interfacing nRF24L01 transmitter Sensor to Arduino pro MINI

details of the Pinout and connections :

Signal	**Arduino pin for TMRh20 RF24 Library**
GND	GND
VCC	3.3 V
CE	7
CSN	8
SCK	13
MOSI	11
MISO	12
IRQ	-

ARDUINO PRO MINI PROGRAMMING

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
// include nRF24L01 header file

RF24 radio(7, 8); // define CE, CSN pins
const byte address = 0xB00B1E50C3LL;
// define specific pipeline for communicate
int ledPin = 9;        // choose the pin for the LED
int buzzer = 2;        // choose the pin for the buzzer
int inputPin = 5;      // choose the input pin (for PIR sensor)
int pirState = LOW;    // we start, assuming no motion detected
int val = 0;           // variable for reading the pin status
int pirState1 = LOW;

void setup() {
  Serial.begin(9600);
  //serial monitor at 9600 baud rate
  pinMode(ledPin, OUTPUT);   // declare LED as output
  pinMode(buzzer, OUTPUT);   // declare buzzer as output
  pinMode(inputPin, INPUT);  // declare sensor as input
  radio.begin();             //start communication
  radio.openWritingPipe(address);
  //start communcating data at defined pipeline
  radio.setPALevel(RF24_PA_HIGH); //set range of the Tx
  radio.stopListening();  // start transmitting data

}

void loop() {
  val = digitalRead(inputPin);  // read input value of pir
  if (val == HIGH) {            // check if the input is HIGH
    digitalWrite(ledPin, HIGH); // turn LED ON
    tone(buzzer, 2940, 1500);
    // turn on the buzzer at 2940Hz & for 1500 ms.
    if (pirState == LOW) {
      pirState = HIGH;
      radio.write(&pirState, sizeof(pirState));
      delay(6000); // pirstate is HIGH for 6 sec.
      pirState = LOW;
      radio.write(&pirState, sizeof(pirState));
      delay(2000); // delay 2 sec.
    }
  } else {
    digitalWrite(ledPin, LOW); // turn LED OFF
    noTone(buzzer);            // turn buzzer OFF
    if (pirState == HIGH){
      pirState = LOW;
    }
  }
}
```

ADVANTAGES

- The given system is handy and portable, and thus can be easily carried from one place to another.
- The circuitry is not that complicated and thus can be easily troubleshooted.

DISADVANTAGES

- The given alarm system determines the presence of the intruder only, and does not determine how many persons are in there actually.
- The motion is detected only when the person cuts through the line of the PIR sensor. So, high performance industrial grade pir sensor is recommended.
- nRF24L01 is good to transmit large amount of data, but, it has power stability issue for long range communication.

POSSIBILITIES

- Bigger PIR sensor can be used for better sensitivity.
- The data transmission link should be more reliable. Thus, I prefer to use more stable components and communication protocol like Xbee.

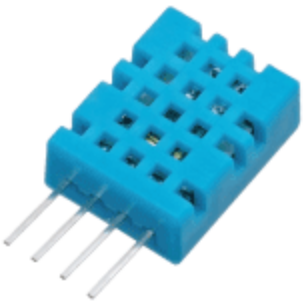
APLLICATIONS

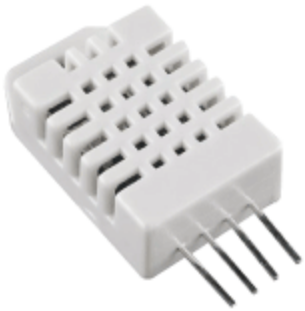
This type of motion sensing alarm and appliance control system can be easily employable for security purposes at banks, various offices and even for sensitive establishments such as for military. Also, We can easily set up this system for household purposes.

4th Module (Temperature sensing system)

In this Arduino Tutorial we will learn how to use the DHT11 or the DHT22 sensor for measuring temperature and humidity with the Arduino board. These sensors are very popular for electronics hobbyists because there are very cheap but still providing great performance. Here are the main specifications and differences between these two sensors:

The DHT22 is the more expensive version which obviously has better specifications. Its temperature measuring range is from -40 to +125 degrees Celsius with +-0.5 degrees accuracy, while the DHT11 temperature range is from 0 to 50 degrees Celsius with +-2 degrees accuracy. Also the DHT22 sensor has better humidity measuring range, from 0 to 100% with 2-5% accuracy, while the DHT11 humidity range is from 20 to 80% with 5% accuracy.





DHT11 ***DHT22***

0 - 50°C / ± 2°C	*Temperature Range*	-40 - 125 °C / ± 0.5 °C
20 - 80% / ± 5%	*Humidity Range*	0 - 100 % / ± 2-5%
1Hz (one reading every second)	*Sampling Rate*	0.5 Hz (one reading every two seconds)
15.5mm x 12mm x 5.5mm	*Body Size*	15.1mm x 25mm x 7.7mm
3 - 5V	*Operating Voltage*	3 - 5V
2.5mA	*Max Current During Measuring*	2.5mA

fig. DHT22 & DHT11 side by side comparison

There are two specification where the DHT11 is better than the DHT22. That's the sampling rate which for the DHT11 is 1Hz or one reading every second, while the DHT22 sampling rate is 0.5Hz or one reading every two seconds and also the DHT11 has smaller body size. The operating voltage of both sensors is from 3 to 5 volts, while the max current used when measuring is 2.5mA.

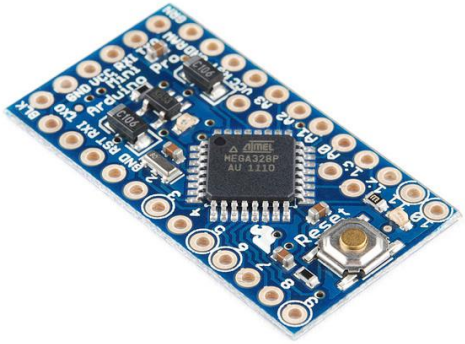
OBJECTIVE

- Detect temperature using DHT22 Sensor and Arduino Pro MINI.
- Setup an Wi-Fi and IoT (Internet of Things) based Alert Mechanism using ESP-12E Wi-Fi Module.
- Send temperature update messages to specified android mobile which are connected via Wi-Fi or Internet.
- Display status in specified android mobile.

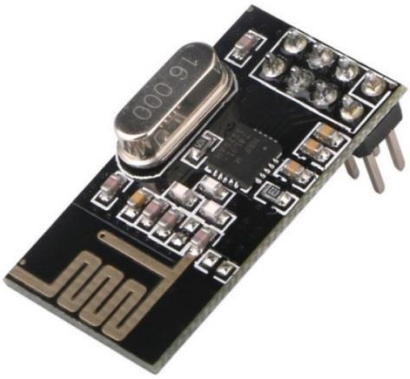
PLANNING OF COMPONENTS

- **DHT22 temperature Sensor** – to sense temperature.(This module can sense humidity also, but , here we are working with only temperature)

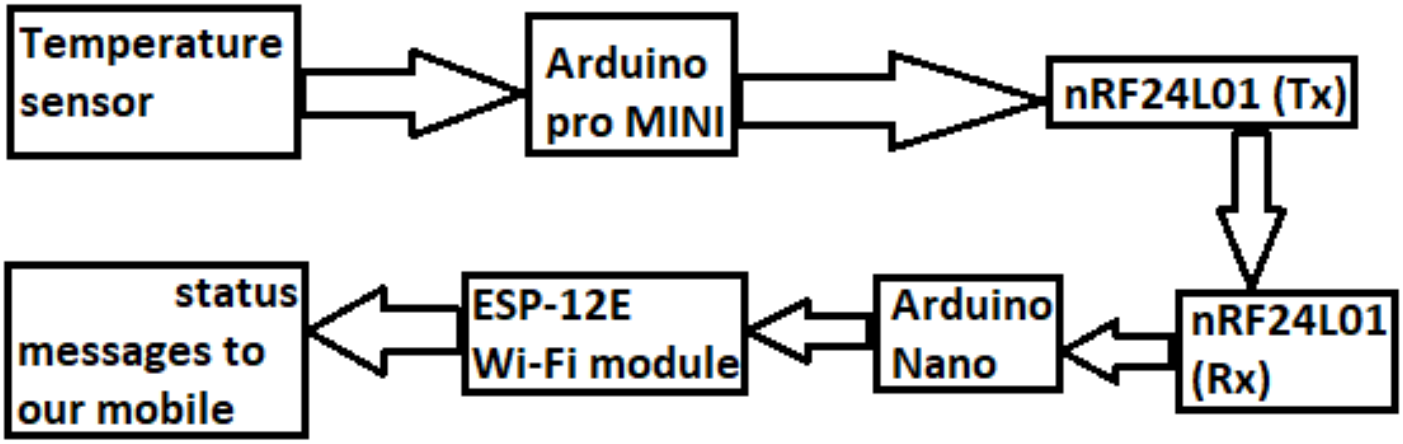
- **Arduino Pro MINI**
  - To read temperature sensor output and detect temperature (through specific digital pin).
  - To send data to Wi-Fi Module.



- **NRF24L01 Module** – for transmitting the data received from the temperature sensor .



BLOCK DIAGRAM



COMPONENT DESCRIPTION

DHT22 temperature sensor

It consists of a humidity sensing component, a NTC temperature sensor (or thermistor) and an IC on the back side of the sensor.

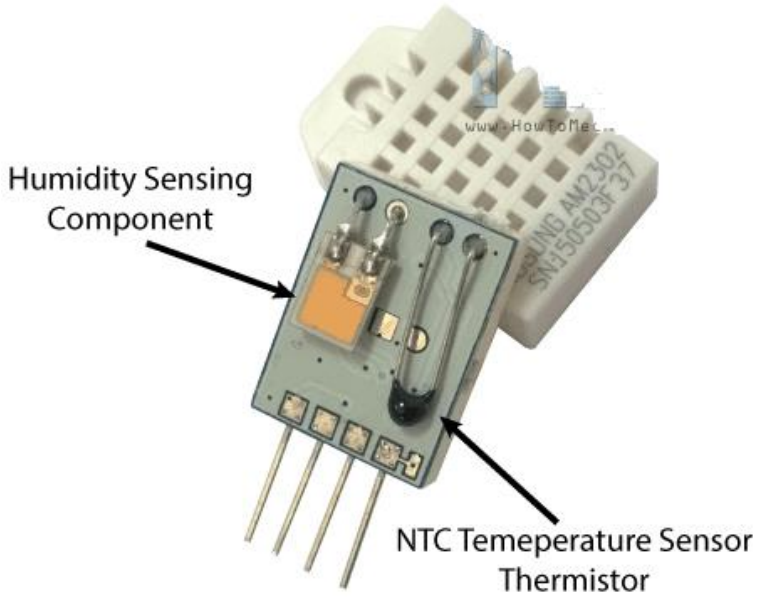


fig. internal components of DHT22

For measuring humidity they use the humidity sensing component which has two electrodes with moisture holding substrate between them. So as the humidity changes, the conductivity of the substrate changes or the resistance between these electrodes changes. This change in resistance is measured and processed by the IC which makes it ready to be read by a microcontroller.

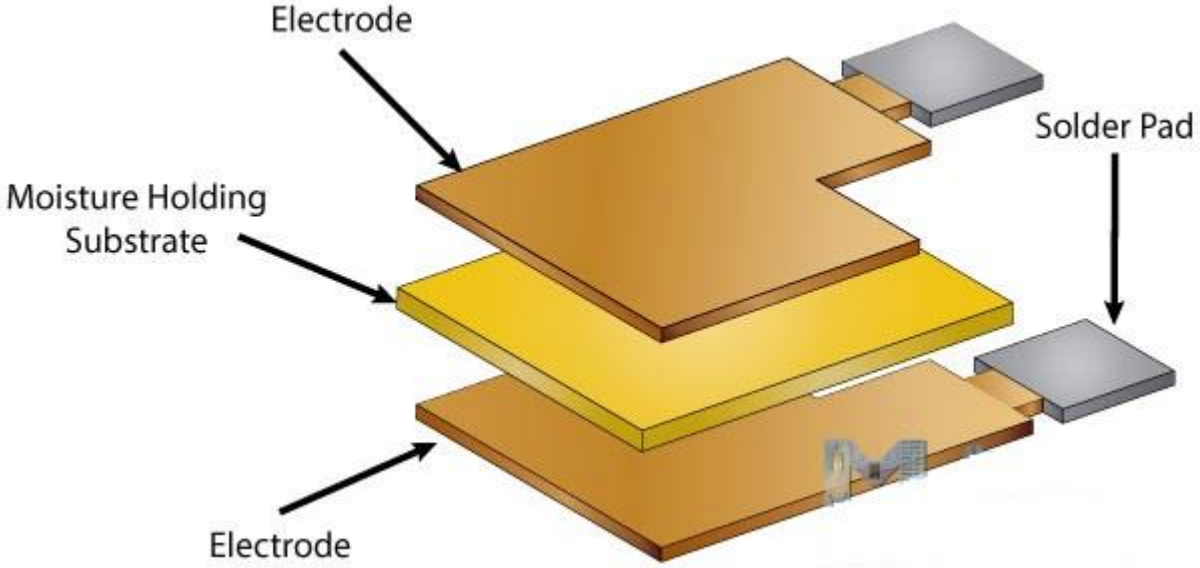


fig. humidity sensing component

On the other hand, for measuring temperature these sensors use a NTC temperature sensor or a thermistor.

A thermistor is actually a variable resistor that changes its resistance with change of the temperature. These sensors are made by sintering of semiconductive materials such as ceramics or polymers in order to provide larger changes in the resistance with just small changes in temperature. The term "NTC" means "Negative Temperature Coefficient", which means that the resistance decreases with increase of the temperature.

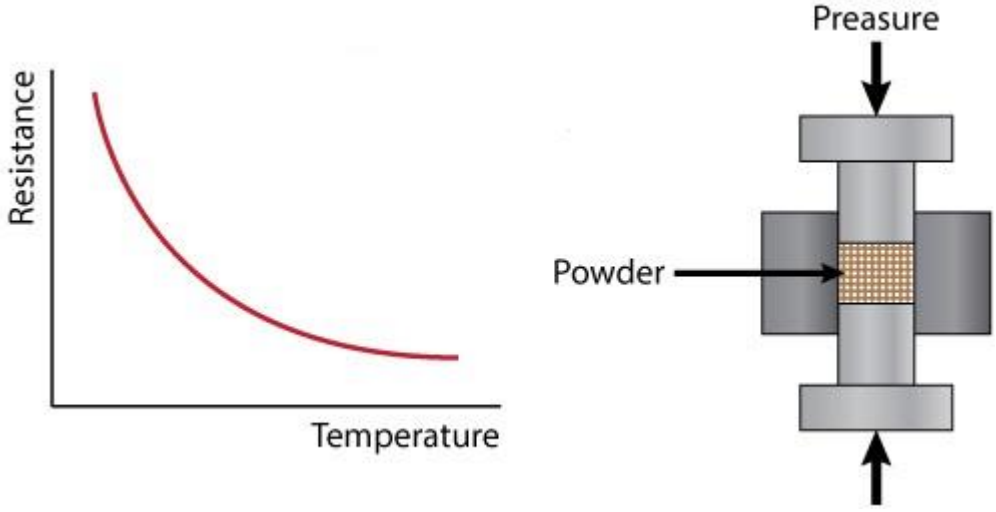


fig. Temperature sensing component

More details about DHT22

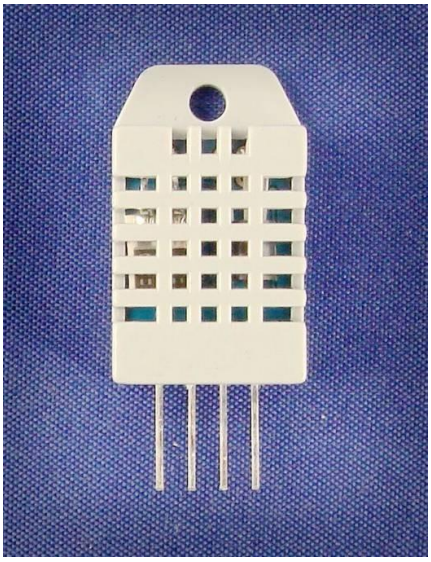


fig. Capacitive-type humidity and temperature module/sensor

1. Feature & Application:

* Full range temperature compensated
* Relative humidity and temperature measurement
* Calibrated digital signal
* Outstanding long-term stability
* Extra components not needed
* Long transmission distance
* Low power consumption
* 4 pins packaged and fully interchangeable

2. Description:

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humiditysensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chipcomputer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and thecalibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will citecoefficient from memory.

Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions.

Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

5. Electrical connection diagram:

Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

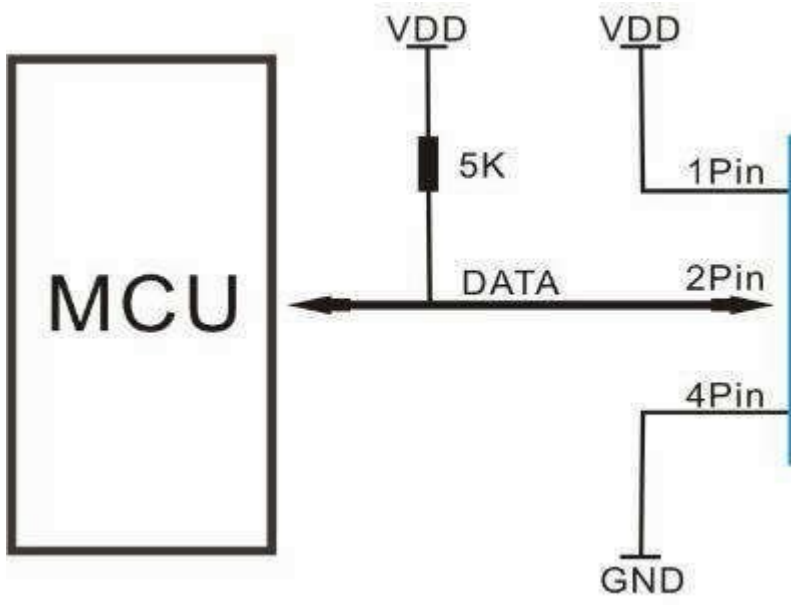


fig. 3Pin---NC, AM2302 is another name for DHT22

6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-6V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

Single-bus data is used for communication between MCU and DHT22, it costs 5mS for single time communication.

Data is comprised of integral and decimal part, the following is the formula for data.

DHT22 send out higher data bit firstly!

DATA=8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data+8 bit check-sum. If the data transmission is right, check-sum should be the last 8 bit of "8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data".

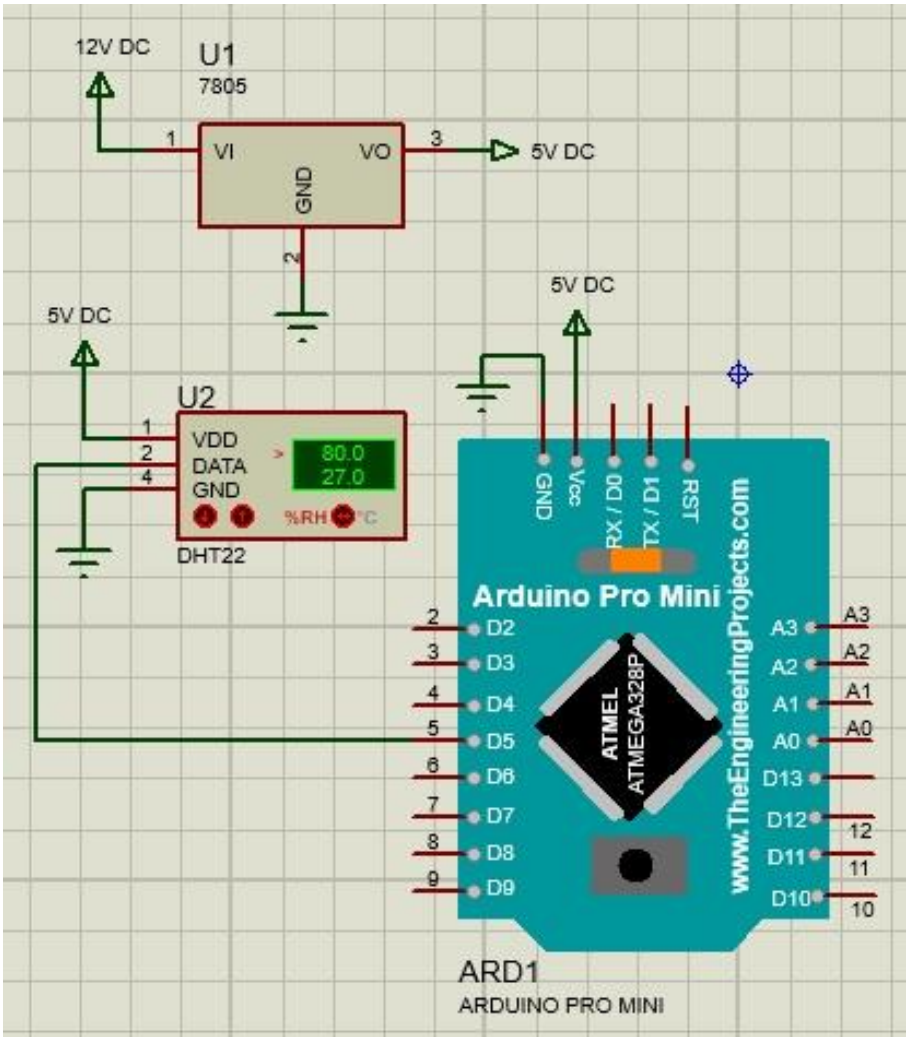
When MCU send start signal, DHT22 change from low-power-consumption-mode to running-mode. When MCU finishes sending the start signal, DHT22 will send response signal of 40-bit data that reflect the relative humidity and temperature information to MCU. Without start signal from MCU, DHT22 will not give response signal to MCU. One start signal for one time's response data that reflect the relative humidity and temperature information from DHT22. DHT22 will change to low-power-consumption-mode when data collecting finish if it don't receive start signal from MCU again.

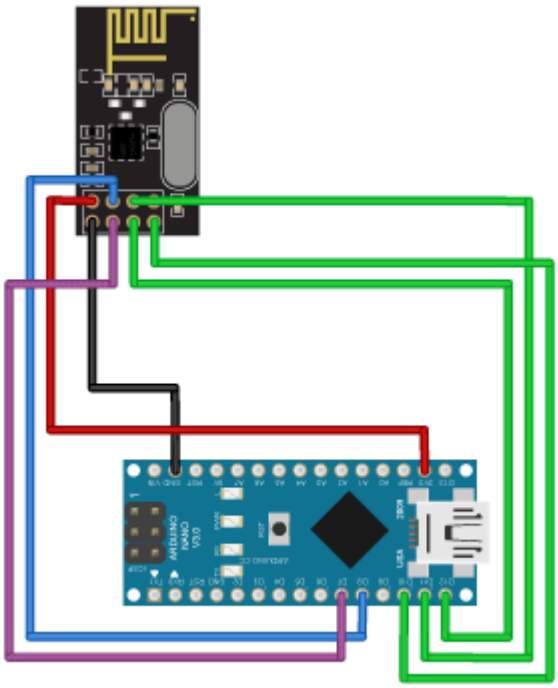
7. Electrical Characteristics:

Item	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

*Collecting period should be : >2 second.

CIRCUIT DIAGRAM





CIRCUIT EXPLANATION

• Interfacing temperature Sensor to Arduino

- Connecting middle pin of DHT22 to digital pin 5 of arduino pro MINI
- Connect Vcc and Gnd pin of DHT22 to the specific ports of arduino pro MINI

• Interfacing nRF24L01 transmitter Sensor to Arduino pro MINI

details of the Pinout and connections :

Signal	Arduino pin for TMRh20 RF24 Library
GND	GND
VCC	3.3 V
CE	7
CSN	8
SCK	13
MOSI	11
MISO	12
IRQ	-

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
// include nRF24L01 header file
#include <dht.h>
#define dataPin 5
/* include DHT22 (temperature sensor) header file
and define in a digital pin */

dht DHT;
float t2 = 0;
RF24 radio(7, 8); // define CE, CSN pins
const byte address = 0xB00B1E50D2LL;
// define specific pipeline for communicate

void setup() {
  Serial.begin(115200);
  //serial monitor at 115200 baud rate
  radio.begin(); //start communication
  radio.openWritingPipe(address);
  //start communcating data at defined pipeline
  radio.setPALevel(RF24_PA_HIGH); //set range of the Tx
  radio.stopListening(); // start transmitting data
}

void loop() {
  int readData = DHT.read22(dataPin);
  // read the i/p data of the datapin of DHT22
  float t1 = DHT.temperature;
  // store the temp. data in a variable
  if ( t1 != t2)
  {
    radio.write(&t1, sizeof(t1));
    Serial.println(t1);
    t2 = t1;
  }
  // temperature will be transmitted when it is changed
  delay(6000); // delay 6 sec.
}
```

ADVANTAGES

- The given system is handy and portable, and thus can be easily carried from one place to another.
- The circuitry is not that complicated and thus can be easily troubleshooted.
- DHT22 compensate full range of temperature.
- calibrated digital signal is provided.
- Long data transmission distance.
- Low power consumption.
- extra components not needed.

DISADVANTAGES

- Humidity sensing is not included in this project.
- nRF24L01 is good to transmit large amount of data, but, it has power stability issue for long range communication.

POSSIBILITIES

- Temperature sensor should be faster and more precise. ex. DHT44
- The data transmission link should be more reliable. Thus, I prefer to use more stable components and communication protocol like Xbee.

APLLICATIONS

This type of temperature sensing system can be easily employable for real time temperature update purpose at anywhere. due to IoT involvement, you can get the temperature of an particular area from anywhere.

- ***Software Design***

The software design part is done by MIT App Inventor and the database which is used in this project is firebase by Google. So, the design part is nothing but designing of graphical user interface (GUI), implementing some button & slider and some notification system on Android application. Using this GUI, user interacts with the system to control devices. For interaction, user initially has to establish connection between Android application and deployed Wi-Fi network. On successful establishment of connection, user can either operate devices (ON/OFF/control) or acquire status information about devices. Further, temperature reading will be constantly notified to user via Android application. In case of emergency situations like gas leakage, user will be given immediate notification in the form of simple notification in the app. When PIR sensors will detect that intruder try to enter in the room, it will automatically turn ON alarm, and security information will be sent to Android Application.

fig. Login Screen

This is the first screen that will be displayed to the user wherein the user has to establish connection with Wi-Fi network for remotely accessing appliances.

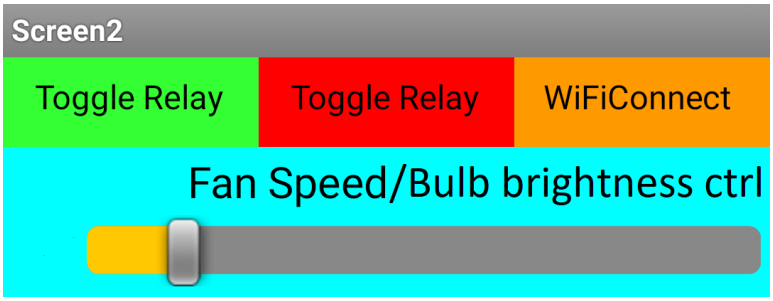


fig. control from application

This allows user to individually turning ON/OFF all the appliances present in a particular area and you can also control the brightness of a bulb or, change the speed of a fan.

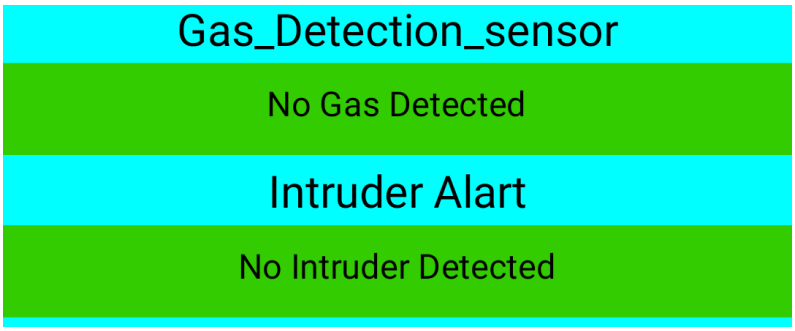


fig. Alert section

This shows an alert message which is "Gas is detected" sent to the users' Android application when LPG leakage occurs. When PIR sensor detects that any movement of someone, alarm present there will automatically be turned ON and this is notified on Android application that "Intruder detected".

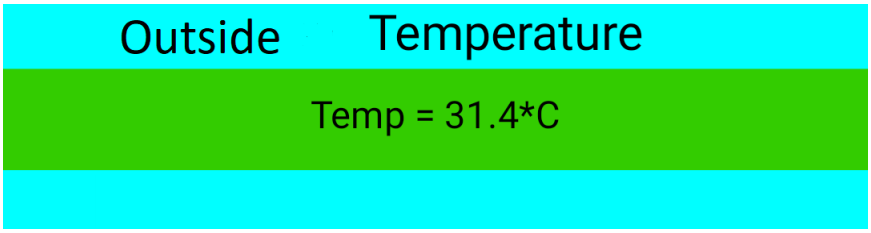


fig. Temperature Display section

Temperature sensor detect the temperature outside of the house and show the temperature through the application.

C. WORKING

The user can access this system using Android handset. User sends command through Android phone whose signal is given to PC via Wi-Fi.

Server machine transmits this command to Wi-Fi chip which converts the digital signal to analog. It consists of ON/OFF AC(alternate current) appliances and controlling them. In other case, wifi chip sends data from 3 modules to the server and the server then update those data in our mobile application. It consists of temperature sensor (DHT22), LPG sensor (MQ6), Relays and PIR sensor. Main circuit in turn will perform the required operation as per the command received. In case if motion is detected or on gas leakage & temperature notification will be received by the user on his handset. ESP12E wifi module transmit and receive data simultaneously, just like router. A router receives internet via Ethernet cable and transmit the internet service to all the direction wirelessly. Here, our ESP12E receives all the data from other 3 modules via arduino nano send it to the database via internet and the database then force the application to update those data. And when we tap anything in our corresponding application like toggling relays or, controlling brightness of a bulb then the application sends some specific data to the database and database force the ESP12E to do the respective job via internet. To this, first we have to supply internet from a device and the device's wifi hotspot should be turned on. Now our job is to connect ESP12E to the internet. The ESP12E automatically connect to our device's hotspot because the ip address of our device is hard-coded. That means ESP12E now connected to the internet. If there is many device , then the code should be dynamic , then ssid (name of the network) and password of the respective device should given to the mobile application login screen and continue to the the next page of the application.

APPLICATIONS

Home automation system has the capabilities to control the following components in users home and monitor the following alarms:

- Temperature and humidity
- Motion detection
- Smoke detection
- Door status
- Light level

Home automation system can control the following appliance:

- Lights on/off/dim
- HVAC on/off
- Window open/close
- On/off different appliance

We can build cross platform system that can be deployed on various platforms like iOS, Windows etc. Home Automation can be extended to all other home appliances. Security cameras can be controlled, allowing the user to observe activity around a house. Security systems can include motion sensors that will detect any kind of unauthorized movement and notify the user. Scope of this project can be expanded to Industries, Hospitals etc. and thus not restricted to home.

COST OF THE PROJECT

Component Name	Quantity	Cost (in rupees)
PCB & veroboard	1 + 1	50
Buzzer	2	20
Gas Sensor	1	150
Servo Motor	1	120
nRF24L01 Trans-receiver	4	320
PIR sensor	1	100
Arduino Board	4	680
solder		20
Temperature sensor	1	350
3.3V power supply module	3	90
5V regulator with heat sink	2	30
12V,2A power supply	2	230
ESP12E	1	220
Optocoupler and PIC	2 + 1	100
Triac	1	10
Transistor, resistors and capacitors		20
		Total = 2510

CONCLUSION

This paper proposes a low cost, secure, ubiquitously accessible, remotely controlled solution for home automation. The approach discussed in the paper is novel and has achieved the target to control home appliances remotely using the Wi-Fi technology to connect system parts, satisfying user needs and requirements. Looking at the current scenario we have chosen Android platform so that most of the people can get the benefit.

The technology is easy to use and targeted for people without technical background. This technology also provides great assistance to handicapped and aged old people.The proposed system is better from the scalability and flexibility point of view than the commercially available home automation systems.

REFERENCES

[1] https://www.youtube.com/

[2] http://technav.ieee.org/tag/279/automation

[3]Ahmed ElShafee, KarimAlaaHamed “Design and Implementation of a WiFi Based Home Automation System” Vol:6 2012

http://waset.org/publications/5037/design-and-implementation-of-a-wifi-based-home-automation-system

[4]DeepaliJavale Assistant Professor Dept. of Computer Engg MAEER's MITCOE Pune, India Mohd. Mohsin Student Dept. of ComputerEngg MAEER's MITCOE Pune, India“Home Automation and Security System Using Android ADK” Volume 3 (March 2013)

http://www.ijecct.org/v3n2/(382-385)0302M22.pdf

[5] http://www.sunrom.com/p/rf-serial-data-link-uart-2-4ghz

[6] http://www.ti.com/lit/ds/symlink/lm35.pdf

[7] http://www.tech-faq.com/infrared-sensors.html

[8] https://google.com