TECH HYAKU

Wireless Doorbell using Arduino and RF Module

Aim:

To design and implement a wireless doorbell system using Arduino and RF (Radio Frequency) modules

Hardware Required:

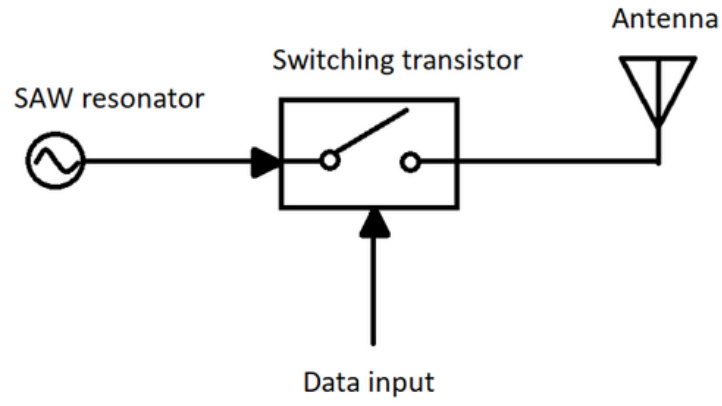
- RF module
- Arduino
- Buzzer
- Push-button
- Breadboard
- Connecting wires

Software Required:

- **Arduino IDE –** For writing and uploading code
- **VirtualWire or RadioHead Library –** To enable RF communication between transmitter and receiver

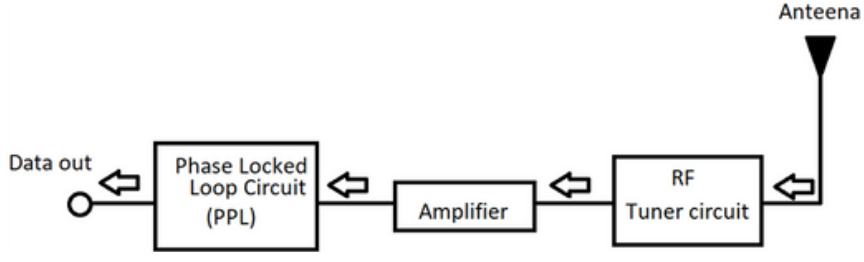
RF Transmitter:

When the input to the data pin is HIGH, the switch will act as a short circuit and the oscillator runs which produces a fixed amplitude carrier wave and a fixed frequency for some period of time‘t’. When the input to the data pin is low, the switch acts as an open-circuit and the output will be zero. This is also known as **Amplitude shift keying (ASK)**.

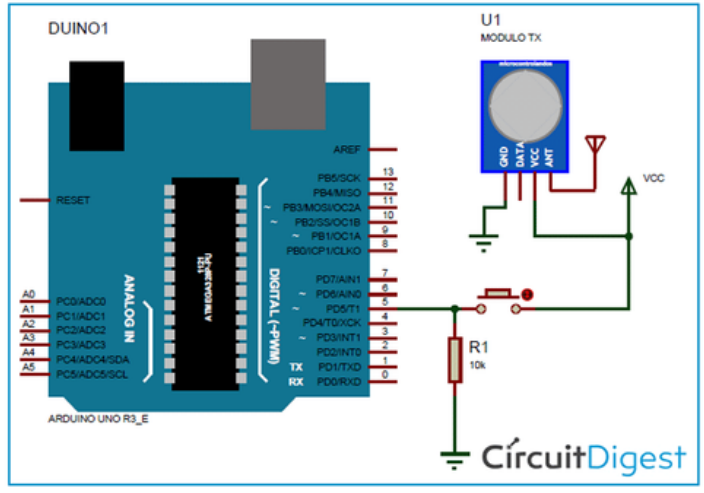


Receiver Circuit:

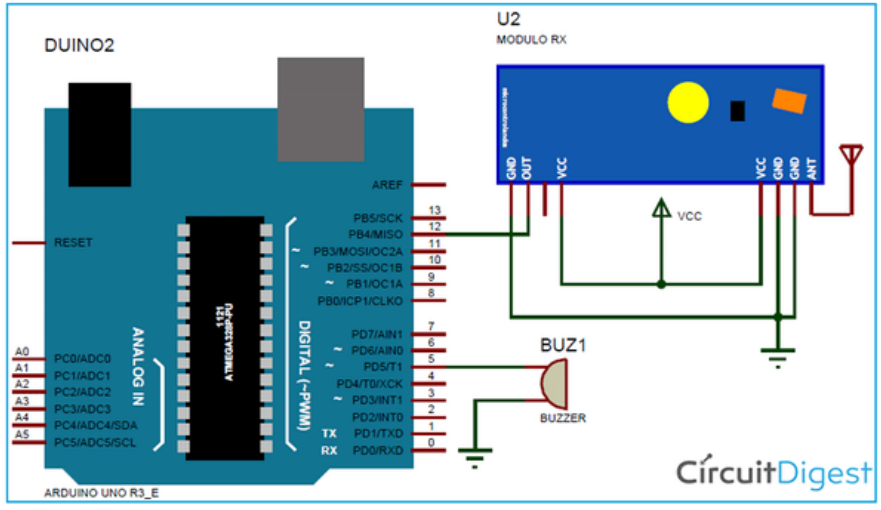
An RF tuner is used to tune the circuit to a particular frequency, which needs to meet the transmitted frequency. An amplifier circuit is used to amplify a particular frequency from all other signals and to increase the sensitivity of the particular frequency.



Arduino RF Transmitter Circuit Diagram



Arduino RF Receiver Circuit Diagram



Complete Project Code

Doorbell Transmitter Code

```
// ask_transmitter.pde
// -*- mode: C++ -*-
// Simple example of how to use RadioHead to transmit messages
// with a simple ASK transmitter in a very simple way.
// Implements a simplex (one-way) transmitter with an TX-C1 module
#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile
RH_ASK driver;
// RH_ASK driver(2000, 2, 4, 5); // ESP8266 or ESP32: do not use pin 11
void setup()
{
   Serial.begin(9600);   // Debugging only
   pinMode(5,INPUT);
   if (!driver.init())
        Serial.println("init failed");
}
void loop()
{
   if(digitalRead(5)==HIGH){
   const char *msg = "a";
   driver.send((uint8_t *)msg, strlen(msg));
   driver.waitPacketSent();
   delay(200);
   }
}
```

Doorbell Receiver Code

```
#include <RH_ASK.h>
#include <SPI.h> // Not actualy used but needed to compile
#include "pitches.h" //add Equivalent frequency for musical note
#include "themes.h" //add Note vale and duration
RH_ASK driver;
void setup()
{
   Serial.begin(9600); // Debugging only
   if (!driver.init())
        Serial.println("init failed");
    else
      Serial.println("done");
}
void Play_Pirates()
{
 for (int thisNote = 0; thisNote < (sizeof(Pirates_note)/sizeof(int)); thisNote++) {
   int noteDuration = 1000 / Pirates_duration[thisNote];//convert duration to time delay
   tone(8, Pirates_note[thisNote], noteDuration);
   int pauseBetweenNotes = noteDuration * 1.05; //Here 1.05 is tempo, increase to play it slower
   delay(pauseBetweenNotes);
   noTone(8); //stop music on pin 8
   }
}
void loop()
{
   uint8_t buf[1];
   uint8_t buflen = sizeof(buf);
   if (driver.recv(buf, &buflen)) // Non-blocking
   {
     Serial.println("Selected -> 'He is a Pirate' ");
     Play_Pirates();
     Serial.println("stop");
 }
}
```

Wireless Arduino Doorbell Working

The transmitter module, along with the Arduino is connected near the door, and the receiver module, along with Arduino, can be installed in any part of the room. When someone presses the switch, it sends the high pulse to the 5th pin of Arduino, which is connected near the door along with the transmitter module. In our Receiver code, we wrote a command- digitalRead(5), this command makes the Arduino, to keep on reading this pin. When this pin gets HIGH, Arduino transmits data through the transmitter, and these signals are received by the receiver. The Arduino, which is connected to a buzzer, reads these signals, and when the desired data is received, the if function is satisfied, and the code will initiate the function, Play_Pirates() and the music will start to play.