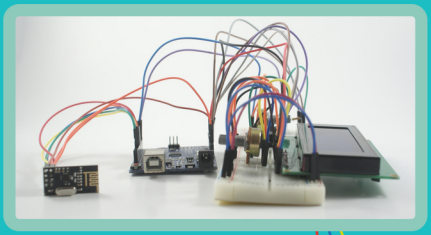
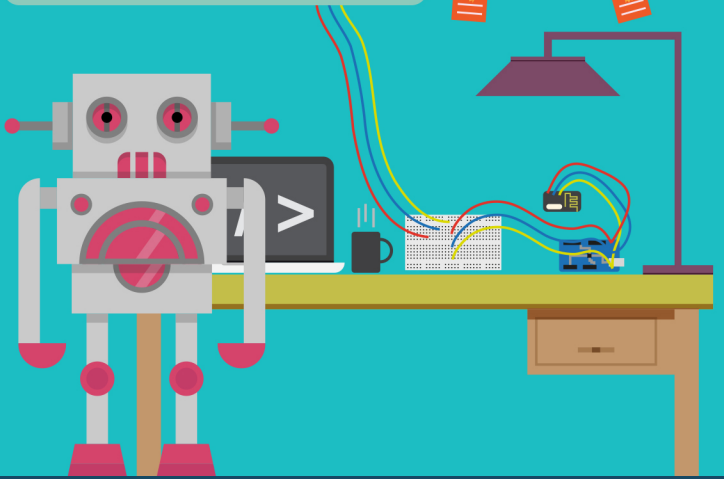
WIRELESS CONTROL CENTER





Creation Crate

BUILDING THE MAKERS OF TOMORROW

INDEX

INTRODUCTION

Welcome to Month 14!

What are we creating?

Control your Month 13 project from across the room!
Introducing this month's build: **Wireless Control Center!**

With this project, you will learn how to use a 2.4GHz radio transceiver to send information about the PIR sensor we used in Month 13

How do we make it?

In two steps:

1. Build the hardware:

We will modify the hardware from last month to add an LCD and the radio receiver. Then, we will use this month's Uno R3 to build a wireless PIR detection system.

2. Programming it:

The program will allow the Uno R3's to communicate with each other. One Uno R3 will continually send the state of its PIR sensor, while the other will receive this information and display it on the LCD screen.

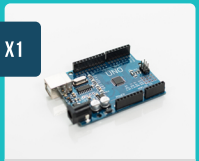
Support Page

https://mycreationcrate.com/month-14

KKQM73

PART LIST

X1

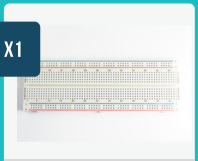


UNO R3

X1

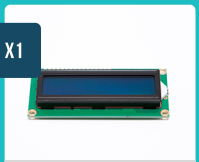
USB CABLE

X1



BREADBOARD

X1

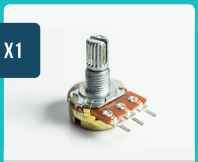


LCD

X2

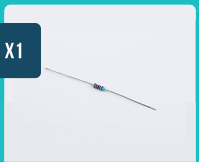
TRANSCEIVER

X1



10K POTENTIOMETER

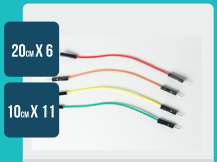
X1



330 OHM RESISTOR

X16

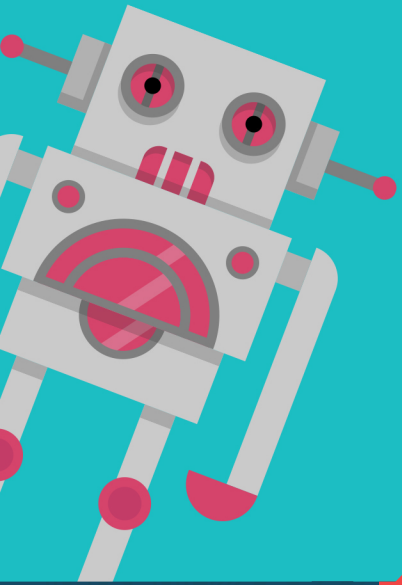
JUMPER WIRE MALE-TO-FEMALE



JUMPER WIRE

RADIO TRANSCEIVER

- This project utilizes the NRF radio transceiver to allow the Uno R3's to communicate to each other. Using the RF24 library, we can call specific radio functions to send and receive data. Because these radios only use a small amount of power, they can only transmit so far. Depending on factors such as interference, weather, and how many walls the signal has to travel through, the radios will most likely get a range between 30 and 50 meters. Past this range, only some of the messages will be able to get through. This may be enough to alert the control center, but could also result in the control center not knowing that the PIR sensor was activated. So when deciding where to place these projects, you may want to keep them only a room or two apart.

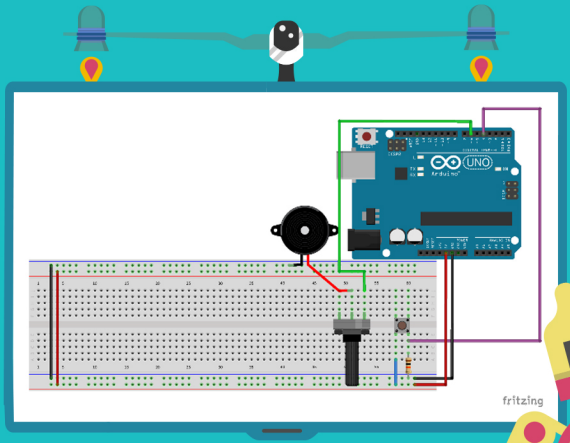


RADIO LIBRARY

- In order to use this code, install the “RF24” library using the Arduino library manager under Sketch>Include Library>Manage Libraries...

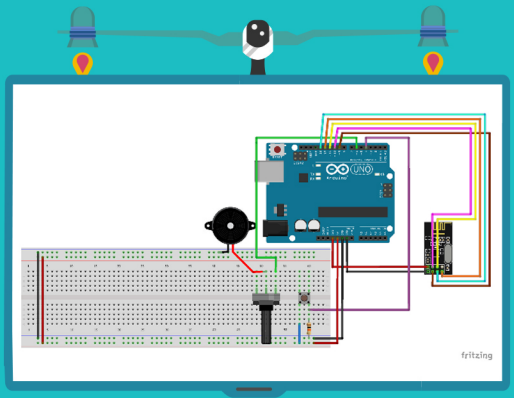
HARDWARE - CONTROL CENTER

- This month's hardware builds off of last month's project. But first, we have to remove some unnecessary components. Take Month 13 and remove the LEDs, LED resistors, and PIR sensor as shown. Also, you may want to move the buzzer and potentiometer over to make room for the LCD.

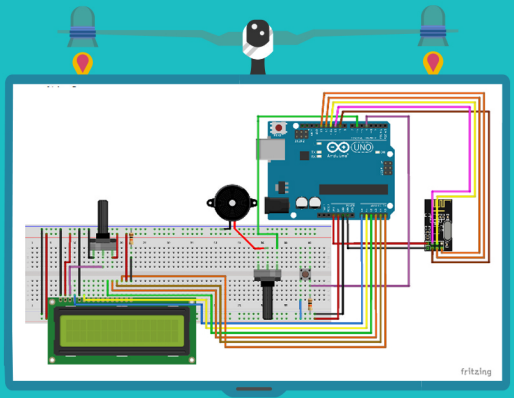


HARDWARE - CONTROL CENTER

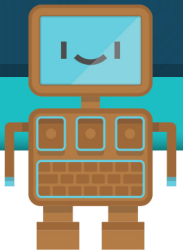
- Now wire the radio:



- Lastly, plug the LCD into the breadboard and wire it as shown:

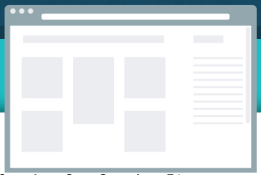


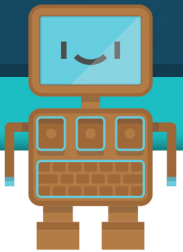
PROGRAM - CONTROL CENTER



```
//Month 14 Control Center

#include <LiquidCrystal.h>
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

//Pin definitions
int buttonPin = 4;
int buzzerPin = 6;

//Security system variables
boolean armed = false;
boolean buzzer = false;
boolean buttonPushed = false;
boolean lostConnection = true;

const uint64_t pipe = 0xE8E8F0F0E1LL; //Address of the radio

int messageCount = 0;
int lostConnectionCount = 0;

double packetLoss = 0; //Percent "lost connection" messages

boolean message[1]; //Holds the read radio message
```

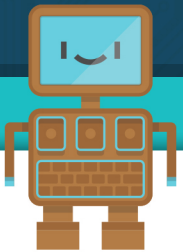
PROGRAM - CONTROL CENTER





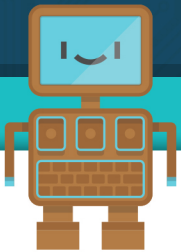
```
LiquidCrystal lcd(A0,A1,A2,A3,A4,A5);
//Creates the lcd object
RF24 radio(9,10);

void updateLCD() //Updates the LCD every 0.5 seconds
{
  if(millis()%500 < 50) //We use < 50 just in case this
function isn't called when millis()%500 is 0
  {
    lcd.clear();
    lcd.print("Status: ");
    if(armed) //Armed/disarmed status
    {
      lcd.print("ARMED");
    }
    else
    {
      lcd.print("DISARMED");
    }
    lcd.setCursor(0,1); //Line 2
    lcd.print("PIR: ");

    if(buzzer)//PIR Status
    {
      lcd.print("DETECTED");
    }
    else
    {
      if(packetLoss < 0.3)
```

PROGRAM - CONTROL CENTER



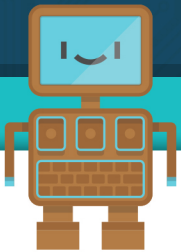
```
{
 lcd.print("CLEAR");
       }
       else
       {
         lcd.print("NO SIGNAL");
       }
     }
   }
}

void setup()
{
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(1, pipe);
  radio.startListening();

  lcd.begin(20,4);
  lcd.clear();

  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop()
{
  if(digitalRead(buttonPin) && !buttonPushed) //If the button
is pushed
  {
```

PROGRAM - CONTROL CENTER



```
 buttonPushed = true; //This is used so the "ready" state is
only toggled once while the button is on
    buzzer = false; //Turn off buzzer if it is on
    armed = !armed; //Toggles the "armed" state
  }
  else if(!digitalRead(buttonPin))
  {
    buttonPushed = false; //When the button is released, allow
it to be pressed again
  }


  if(buzzer) //Turn on the buzzer
  {
    if(millis()%1000 < 500) //Between 0-499 milliseconds
    {
      tone(buzzerPin, 880); //Plays an 'A'
    }
    else //Between 500-999 milliseconds
    {
      tone(buzzerPin, 659); //Plays an 'E'
    }
  }
  else
    noTone(buzzerPin); //Play no sound
```

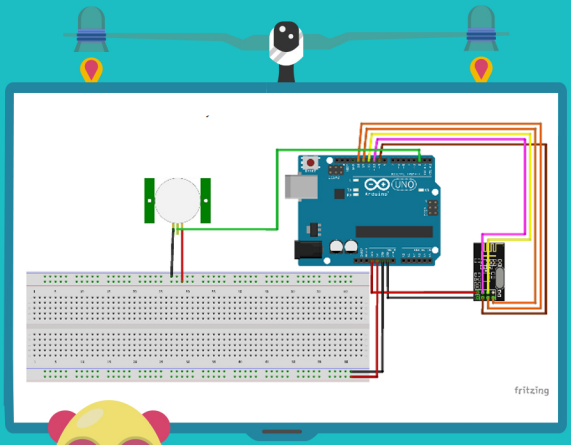
PROGRAM - CONTROL CENTER

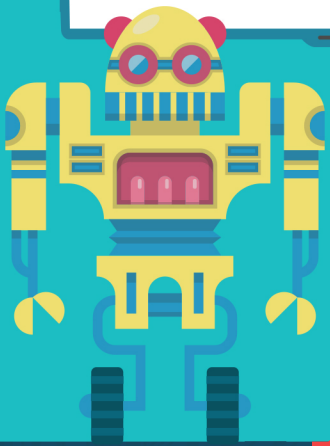


```
if(radio.available()) //Message recieved
  {
    radio.read(message, 1);
    if(message[0] && armed)
      buzzer = true;
    lostConnection = false;
  }
  else
  {
    lostConnection = true;
  }

  //Calculates the percentage of lost messages vs messages
  packetLoss = lostConnectionCount / (double)(messageCount +
lostConnectionCount);

  if(lostConnection)
  {
    lostConnectionCount++;
  }
  else
  {
    messageCount++;
  }

  if((lostConnectionCount + messageCount) > 100) //Resets the messages
counter once enough have been recieved
  {
    lostConnectionCount = 0;
    messageCount = 0;
  }

  updateLCD();
  delay(30); //Gives a second for the radio to catch up
}
```

HARDWARE - PIR TRANSMITTER

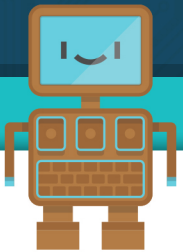
Once you've built and programmed the Control Center, unplug it and start to build the PIR Transmitter. Take the PIR sensor you removed earlier and use it here:





PROGRAM - PIR TRANSMITTER

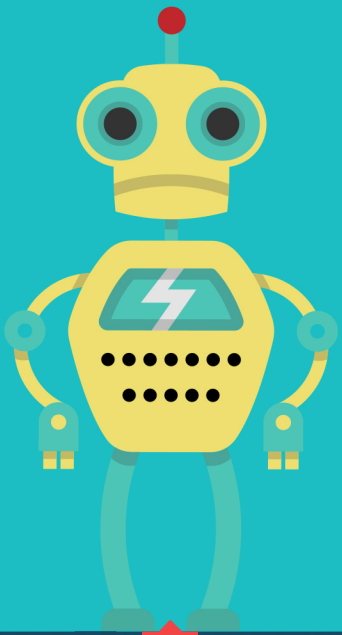
NOTE:
Make sure you unplug the Control Center before uploading, or you might upload to the wrong Uno R3 board!



```
//Month 14 PIR Transmitter

#include  <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

bool message[1];

RF24 radio(9,10);
const uint64_t pipe = 0xE8E8F0F0E1LL;

//Pin definitions
int infraredPin = 2;

void setup()
{
  Serial.begin(9600);
  pinMode(infraredPin, INPUT);

  radio.begin();
  radio.openWritingPipe(pipe);
}
void loop()
{
  Serial.println(digitalRead(infraredPin));
  message[0] = digitalRead(infraredPin);
  radio.write(message,1);
}
```

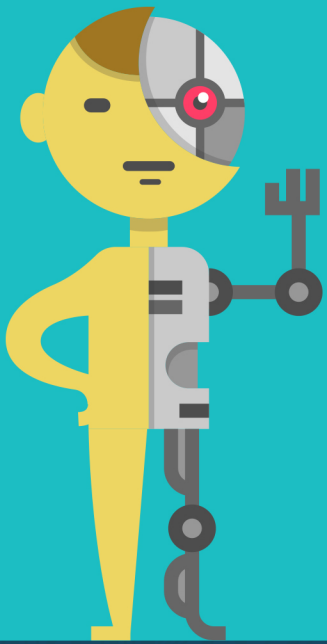
WHAT YOU SHOULD SEE

- When you power up the Control Center, the LCD should light up. The top line of the LCD indicates whether or not the system is armed. If it isn't armed, it will ignore the PIR sensor being triggered. The second line shows the state of the PIR signal. If the PIR transmitter is off, the LCD should say **PIR: NO SIGNAL**. If the PIR transmitter is on and nearby, the LCD should say **PIR: CLEAR** or **PIR: DETECTED**.



MONTHLY CHALLENGE

- Instead of sending a true or false value, send two different integer values depending on the state of the PIR sensor.
  Hint: You have to change `boolean message[1]` to `int message[1]`.



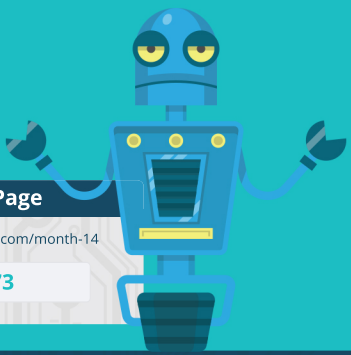
EXERCISES

Solve these problems and write the answers below.

1.) Display the packetLoss variable on the LCD screen.
Answer:

2.) Have the LCD display the total number of times the PIR sensor has been triggered.
Answer:

3.) Make the buzzer play a short beep if the Control Center loses connection to the PIR Transmitter for more than 5 seconds.
Answer:



Support Page

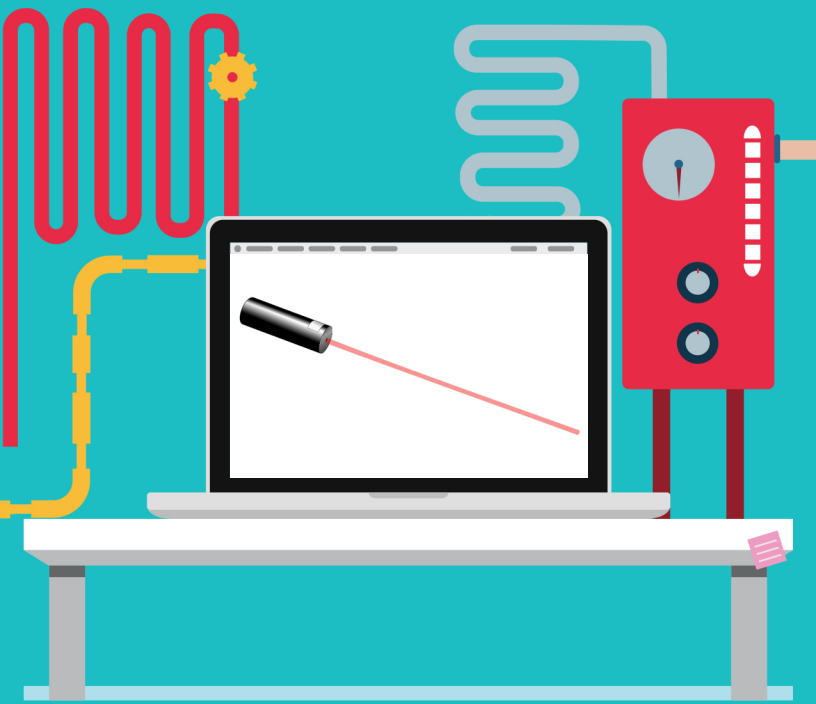
https://mycreationcrate.com/month-14

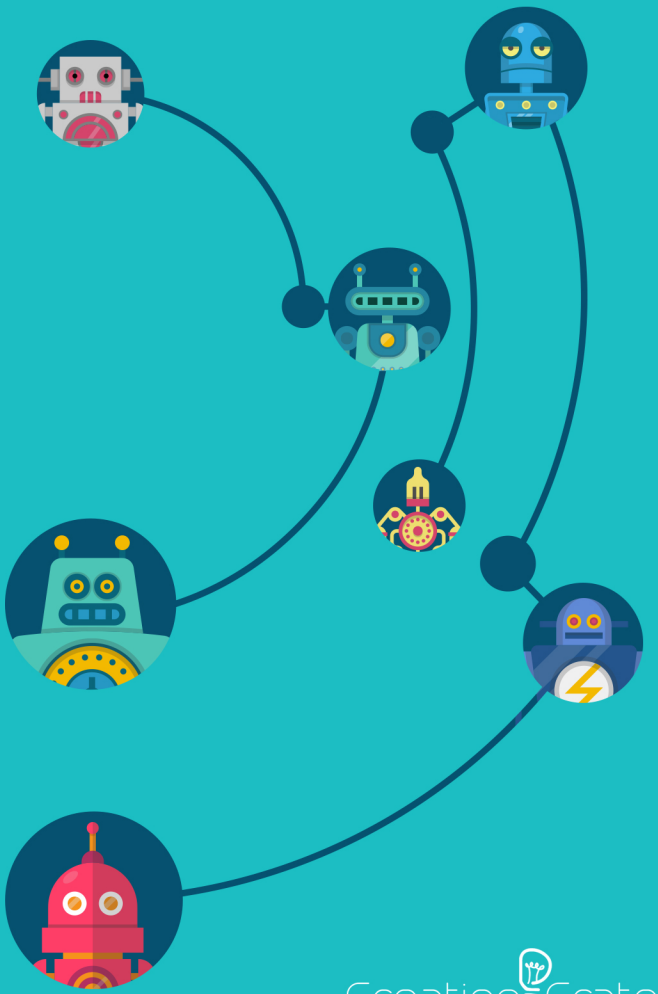
KKQM73

SNEAK PEEK

Here's a sneak peek at next month's project!

Next month's project will build upon this month's project, so make sure you keep all the parts together until next month! Can you guess what it is?





Creation Crate

BUILDING THE MAKERS OF TOMORROW