Advanced Arduino Lab - Sensors

GNG1103 - Engineering Design

Objective

The objective of this lab is to teach students how to utilize the Arduino IDE Software platform to write logic commands that respond to an input from a sensor. This code will be used to control inputs and outputs of an Arduino Uno microcontroller. This lab builds on the content of the "Arduino Programming and Building Lab" and assumes the students have a basic understanding of how to use Arduino. Please refer back to the "Arduino Programming and Building Lab" when needed.

Downloads

The Arduino IDE can be downloaded from here: https://www.arduino.cc/en/Main/Software

The library for the Ultrasonic Range Finder:

NewPing.h (https://playground.arduino.cc/Code/NewPing#Download)

Apparatus and Equipment Overview

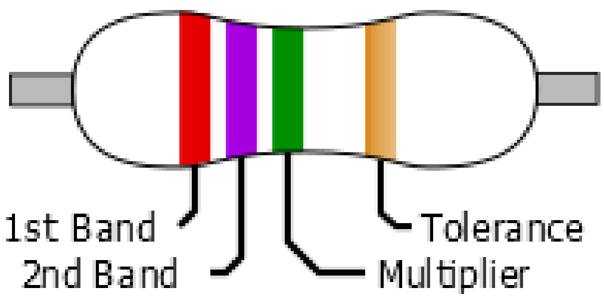
The equipment that will be used in this lab includes:

- 1 × Arduino Uno R3 microcontroller
- 1 × USB 2.0 type A plug to USB 2.0 type B plug cable
- 1 × Lab or personal computer (running Windows, Macintosh or Linux OS)
- 1 × Arduino IDE software
- 1 × Photoresistor
- 1 × Button
- 1 × 5mm LED
- 1 × 330Ω resistor
- 2 × 10kΩ resistor
- 1 × Ultrasonic range finder
- 1 × Passive Infrared motion sensor
- 9 × Lengths of wire
- 1 × Piezo buzzer

Background

Resistor

A resistor is an electrical component which creates electrical impedance, or resistance to current flow. The amount of resistance a resistor provides can be read from the bands of colour on the resistor, which are read left to right. For four bands resistors, the first and second bands represent digits, while the third band represents a multiplier to multiply the digits of the first and second band by. The fourth band is the tolerance, it represents by how much the resistor may deviate from the value indicated by the bands. The value of the resistor can also be determined using the ohmic function of a multimeter. The table and image below shows how to read the colours.



For example, this resistor has a value of 2700 kΩ and a tolerance of ± 5%.

Red	Purple	Green	Gold
2	7	100 kΩ	±5%

27 × 100 kΩ ±5%
= 2700 kΩ ±5%

Colour	1st Band	2nd Band	Multiplier	Tolerance
Black	0	0	1Ω	
Brown	1	1	10Ω	± 1%
Red	2	2	100Ω	± 2%
Orange	3	3	1kΩ	
Yellow	4	4	10kΩ	
Green	5	5	100kΩ	± 0.5%
Blue	6	6	1MΩ	± 0.25%
Violet	7	7	10MΩ	± 0.10%
Grey	8	8	100MΩ	± 0.05%
White	9	9	1GΩ	
Gold			0.1Ω	± 5%
Silver			0.01Ω	± 10%

Resistors are often used in series with components to reduce the amount of current flowing through a circuit, often to protect components rated for lower current amounts.

Pull-up and pull-down resistors are used to bias an input on the Arduino to be either HIGH or LOW respectively. This needs to be done as the resting level of the input isn't necessarily 0. This is especially useful when working with sensors that have an analog output.

Photoresistor

A photoresistor is a resistor that changes its resistance to current in response to the amount of light it is exposed to. As the level of light the resistor is exposed to increases, its resistance decreases, conversely, as the level of light the resistor is exposed to decreases, its resistance increases.

Button

A button is a electrical component that allows electrical current to pass when the button is pressed, otherwise it does not permit electrical current to pass.

Ultrasonic Range Finder

An ultrasonic range finder is a component that measures distance using sound. The component emits a sound at a high frequency, above the range of human hearing, and then listens for the echo of the sound wave. The speed of sound in air is dependent on the atmospheric conditions, such as the temperature, humidity and air pressure, as a result the speed of sound varies from place to place and time to time. Despite the speed of sound through air changing, the ultrasonic range finder can still determine distances fairly accurately. The ultrasonic range finder has a limit to the distance it can measure, from about 3 cm to 400 cm, as well as how often it can measure the distance, approximately every 29 ms.

Passive Infrared Motion Sensor (PIR)

A passive infrared motion sensor is a component that detects changes in the amount of infrared light it receives. When it detects a change in the amount of infrared light it receives, it brings the output pin to HIGH, when there is no change, the output pin is LOW. PIRs are often used in home security systems.

Pre-Lab Preparation

Before arriving to the lab, students should review the lab manual and familiarize themselves with the lab setup and procedure. Furthermore, it is recommended the students review the "Arduino Programming and Building Lab" as this lab assumes the students have a solid understanding of the aforementioned lab's content. It is recommended that students use their own computers for this lab, however lab computers are available. If using a personal computer it is the responsibility of the students to download and install the Arduino IDE as well as all the required libraries, all of which can be found in the downloads section. Furthermore it is encouraged that students review and practice the Arduino syntax beforehand. The Arduino reference can be found here:
https://www.arduino.cc/reference/en/

Pre-Lab Questions

What does each band, from left to right on a resistor represent?

____

____

How does an ultrasonic range finder determine the distance between it and an object?

____

____

Why might a resistor be placed is series with a LED or other electrical component?

____

____

What is the purpose of a pull-down resistor?

____

____

What is the main difference between ultrasonic and PIR sensors?

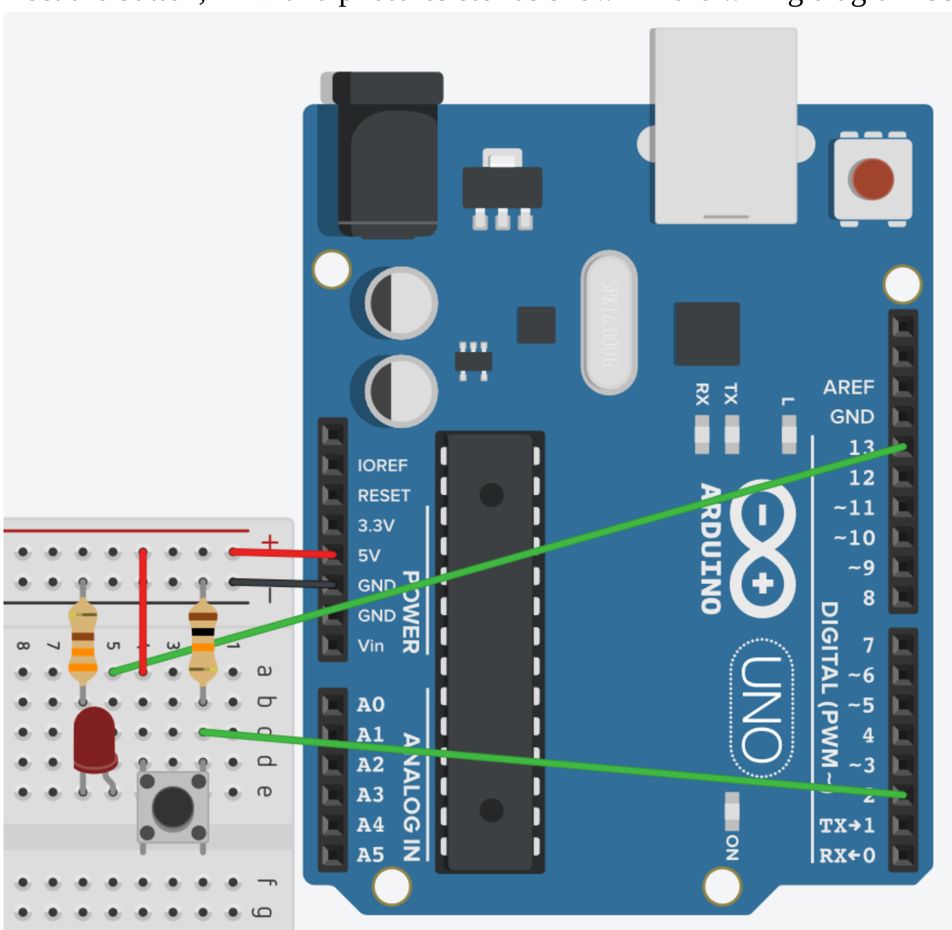
____

____

Part A - Photo Resistor, Button & LED Program

This part of the lab will cover hooking up a photoresistor, button and LED to the Arduino as well as how to create a program to control the LED using the photoresistor and button. The LED will act as an output, while to photo resistor will determine how bright the LED be while the button is pressed.

1. Connect the button, LED and photoresistor as shown in the wiring diagram below.

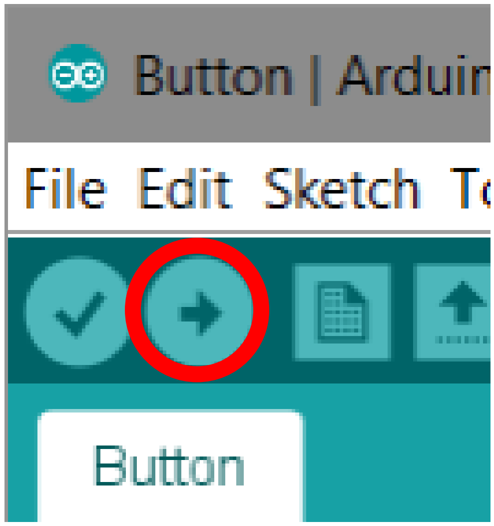


*Note the colours of the resistors, we have a 10kΩ resistor from the output of the button to ground, this is a pulldown resistor. It is used to bias the input pin to LOW.

2. Launch the Arduino IDE software. Once started select Tools>Board>Arduino/GenuinoUno and select the port the Arduino is connected to at Tools>Port. Open File>Examples>02.Digital>Button. This will open a new sketch that looks like the one below.

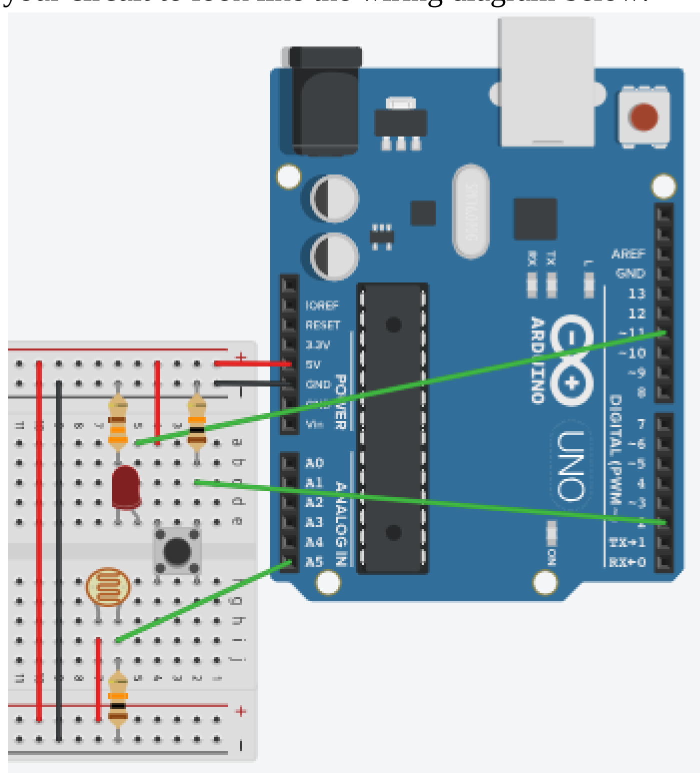
```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

3. Now upload the sketch to the board by clicking the button highlighted in the image below. This will try to compile the program and upload it to the board. If there are any errors in compilation or uploading the Arduino IDE will display the error message at the bottom of the IDE in the console area.



4. Once the upload is complete, the program will automatically run on the microcontroller. In this case, the LED will turn on whenever the button is pressed.

5. Now modify your circuit to look like the wiring diagram below.



*Note that we change the output pin from 13 to 11, this is because we need a analog output. A pulldown resistor was placed on the output of the photoresistor.

6. Now add or modify the following lines of code to the program;
- Modify "const int ledPin = 13" to the new pin of the LED, pin 11.
- On the next line add "const int sensorPin = A5;"

- On the line after “buttonState = 0;” add “int sensorValue;”
- In setup add “Serial.begin(9600);”
- At the beginning of the loop add “sensorValue = analogRead(sensorPin);” and immediately after add “Serial.println(sensorValue);”
- In the if statement, when the buttonState is High, change the digitalWrite to “analogWrite(ledPin, sensorValue);”

The program should now look like the image below.

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  11;      // the number of the LED pin
const int sensorPin = A5;

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int sensorValue;

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    analogWrite(ledPin, sensorValue);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

7. Upload and test out your new program. What happens when you press the button now? Try changing the amount of light the photoresistor is exposed to, what happens?

____

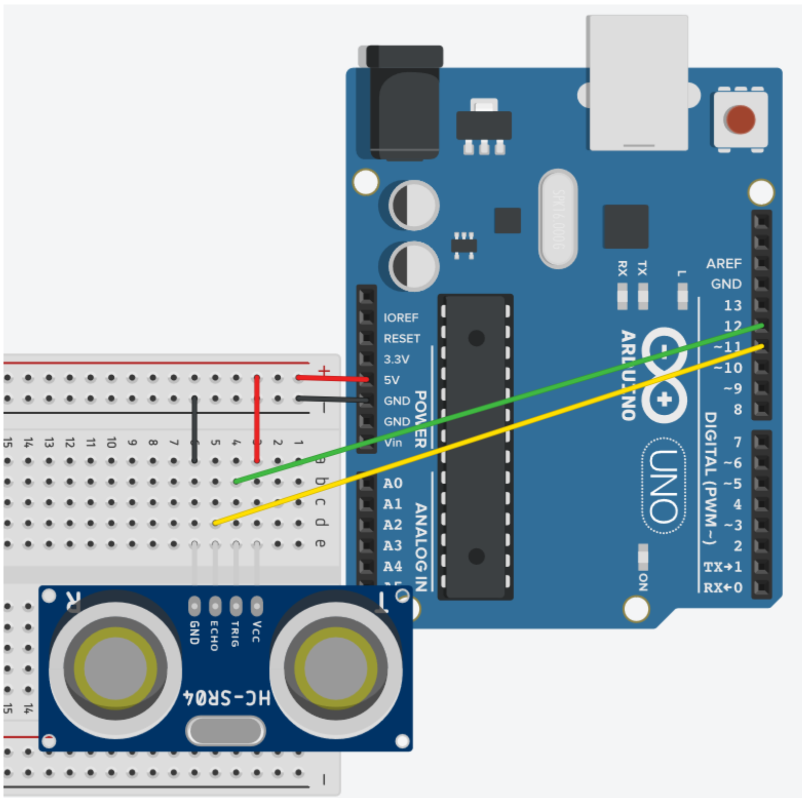
____

8. Think of a way to change the behavior of the LED in response to the input from the photoresistor and implement it!

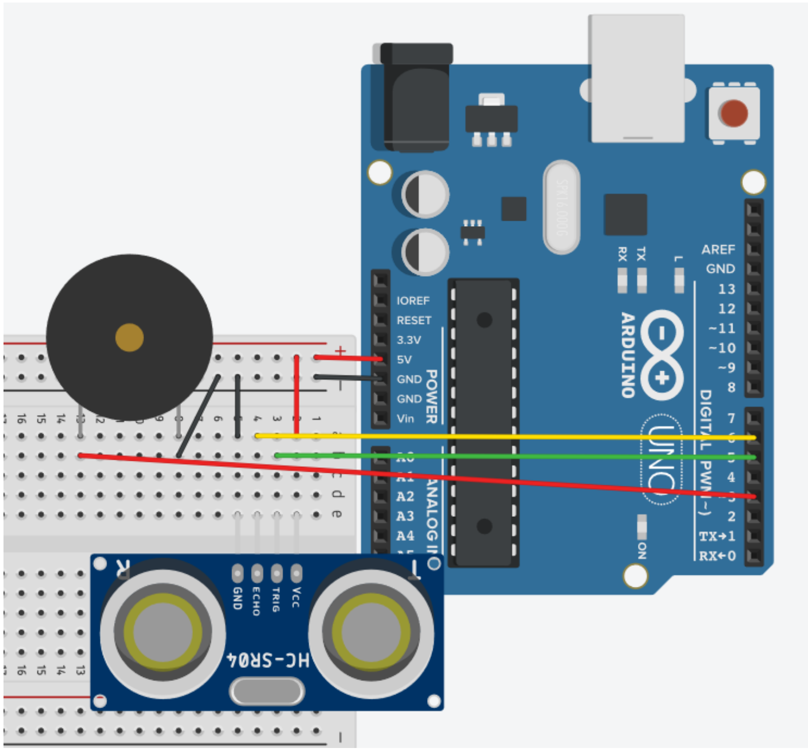
Part B - Ultrasonic Range Finder

In this part of the lab you will create a program that will output the distance the ultrasonic range finder is detecting to the Arduino IDE console. To test the program and circuit you will place your hand or another solid object in front of the ultrasonic range finder and move it back and forth. The console output should indicate the distance to the object fairly accurately. We'll use the output of the ultrasonic range finder to control the volume of a piezo speaker makes.

1. Create the circuit in the wiring diagram below.



2. In the Arduino IDE goto File>Examples>NewPing>NewPingExample, this should open up a sketch in a new window. Find Serial.begin() and change the baud rate to 9600. Upload the program and test it.
3. Rewire the "TRIG" and "ECHO" pins to pins 5 and 6 respectively. Modify the code to reflect the changes made to the wiring, then upload and test.
4. Now we're going to attach a buzzer to the Arduino, as seen in the picture below.

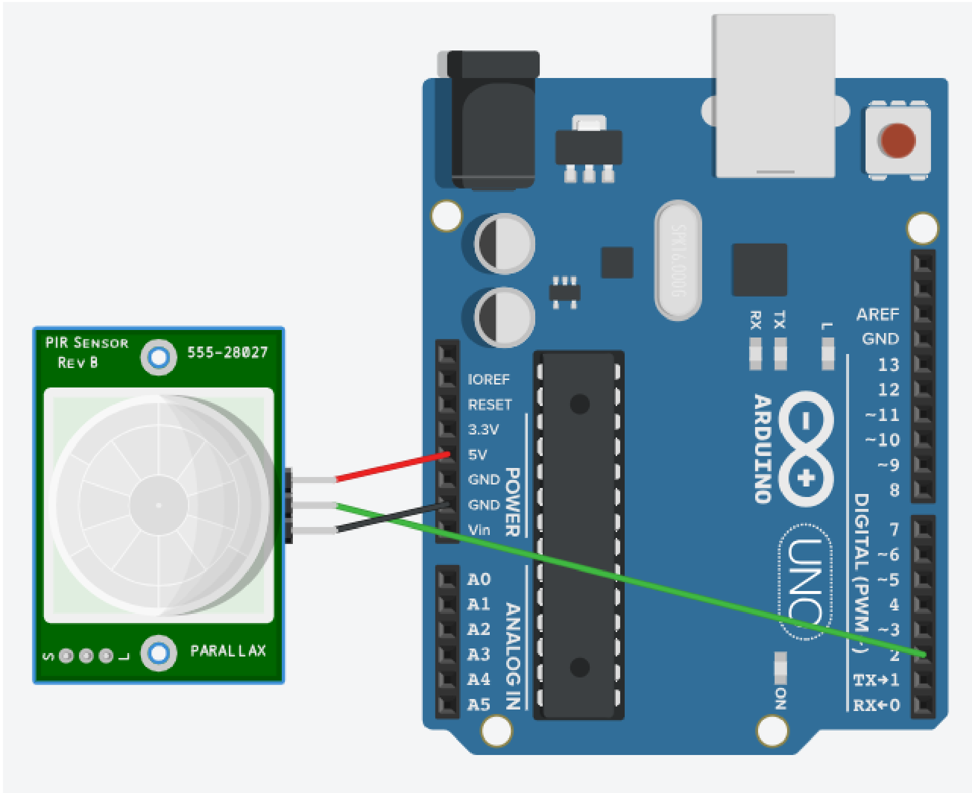


5. Now, we need to modify the code to use the buzzer. Normally, when using a piezo speaker, the tone() function would be used, however this function is incompatible with the NewPing library. As a result we'll need to implement tone generation ourselves. We need the piezo speaker to oscillate from HIGH to LOW at the desired frequency. To do this go get the example in Brightspace called 'URFPiezo2_En.ino'.

Part C - Passive Infrared Sensor

In this section of the lab you will learn how to use a PIR (Passive Infrared Sensor). It is important to note that the sensor has a wide field of view when testing it. This part of the lab will cover the thought process behind writing the code for the PIR. The program that will be made will use the PIR to detect movement, whenever movement is detected, the LED attached to Arduino pin 13 will light up. Additionally, our program will write to the console whenever movement starts and stops. Try to complete each step without looking at the sample code.

1. To begin, hook up the circuit as shown in the image below.



2. To begin writing the program, start a new sketch, go to File>New.
3. We'll begin the program by declaring the variables that will be used, in this case, we need a variable to represent the LED pin, the PIR input pin, weather something has just started moving or is continuing to move. To do this, before the setup() write the following

```
int ledPin = 13;
int pirInput = 2;
bool wasMoving = false;
```

4. When the Arduino first boots up and runs the program, it needs to configure the modes of the various pins the program uses. In this case, the LED pin needs to be an output, and the PIR input pin needs to be an input, so in the setup() block, add the following.

```
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(pirInput, INPUT);
```

5. The Arduino also needs to output text to the Arduino IDE console, so we need to initiate the serial connection. Add the following to the setup() block.

```
  Serial.begin(9600);
```

6. Now that the Arduino has set up all the things the program will be using throughout the main loop, we start working on the main body of the program, the loop. Each time the loop is run, we want to know if the PIR sensor has detected any movement, if there is movement, the program needs to turn the LED on, otherwise the program should turn the LED off. Add the following.

```
void loop() {
  if (digitalRead(pirInput) == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

7. Finally, we want the Arduino to write to the console if the PIR output changes, in other words, only if the PIR just started or stopped detecting movement. In the end the code should look like the following.

```
int ledPin = 13;
int pirInput = 2;
bool wasMoving = false;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(pirInput, INPUT);
  Serial.begin(9600);
}

void loop() {
  if (digitalRead(pirInput) == HIGH) {
    digitalWrite(ledPin, HIGH);
    if(wasMoving == false){
      wasMoving = true;
      Serial.println("Movement started");
    }
  } else {
    digitalWrite(ledPin, LOW);
    if(wasMoving == true){
      wasMoving = false;
      Serial.println("Movement stopped");
    }
  }
}
```

This part of the lab was adapted from the Adafruit PIR sensor tutorial.

8. Now test out the PIR, how sensitive is it? How far or close can it detect movement? How big is its field of view?