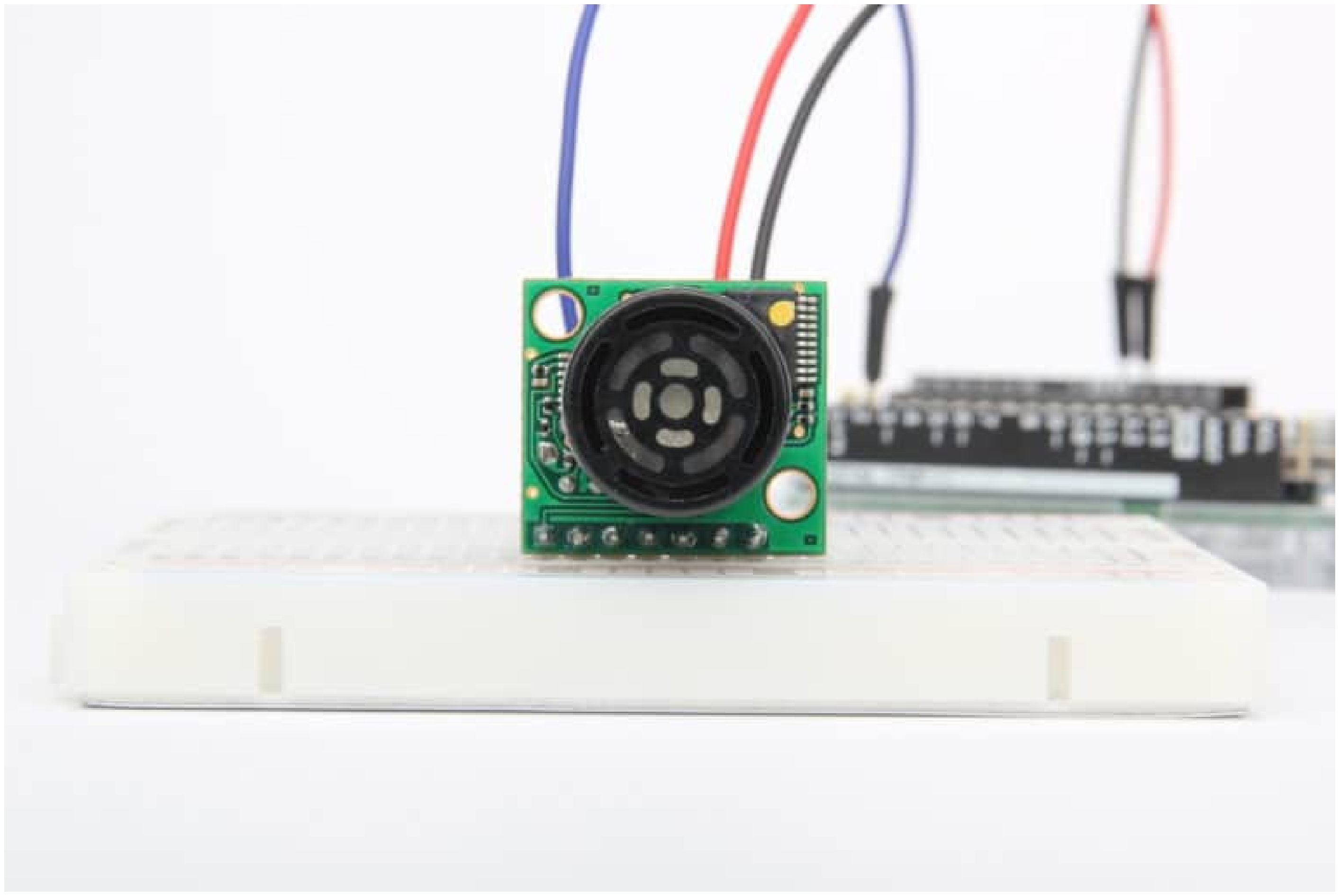
Makerguides.com

MaxBotix MB1240 ultrasonic distance sensor Arduino tutorial

Written by Benne de Bakker (https://www.makerguides.com/author/benne-de-bakker/)



The MB1240 XL-MaxSonar-EZ4 (https://amzn.to/2ZZeYy4) is a high-performance ultrasonic distance sensor with a range of 20 to 765 cm. It is a great alternative for the popular HC-SR04 when you need better noise filtering and reliability. Although this tutorial is written for the MB1240, it can also be used for other MaxBotix sensors.

In this tutorial, you will learn how the sensor works and how you can use it with an Arduino. I have included 3 examples with wiring diagrams that show the basic operation of the sensor. We will look at the different outputs of the sensor and how you can trigger the sensor with a push button.

After each example, I break down and explain how the code works, so you should have no problems modifying it to suit your needs.

If you have any questions, please leave a comment below.

If you would like to learn more about other distance sensors, then the articles below might be useful:

- Waterproof JSN-SR04T Ultrasonic Distance Sensor with Arduino Tutorial (https://www.makerguides.com/jsn-sr04t-arduino-tutorial/)
- How to use a SHARP GP2Y0A21YK0F IR Distance Sensor with Arduino (https://www.makerguides.com/sharp-gp2y0a21yk0f-ir-distance-sensor-arduino-tutorial/)
- How to use a SHARP GP2Y0A710K0F IR Distance Sensor with Arduino (https://www.makerguides.com/sharp-gp2y0a710k0f-ir-distance-sensor-arduino-tutorial/)
- How to use a HC-SR04 Ultrasonic Distance Sensor (https://www.makerguides.com/hc-sr04-arduino-tutorial/)

Supplies

Hardware components

(https://amzn.to/2ZZeYy4)	MB1240 XL-MaxSonar EZ4 (https://amzn.to/2ZZeYy4)	× 1	Amazon (https://amzn.to/2ZZeYy
(https://amzn.to/374aJjX)	Arduino Uno Rev3 (https://amzn.to/374aJjX)	× 1	Amazon (https://amzn.to/374aJjX
	Breadboard (https://amzn.to/2sZTxNA)	× 1	Amazon (https://amzn.to/2sZTxN
	Jumper wires (https://amzn.to/2EG9wDc)	~ 10	Amazon (https://amzn.to/2EG9w
	Header pins (https://amzn.to/2NYud3e) (optional)	× 7	Amazon (https://amzn.to/2NYud3
(https://amzn.to/2QhXhWi)	Momentary push button (https://amzn.to/2QhXhWi)	× 1	Amazon (https://amzn.to/2QhXh
	USB cable type A/B (https://amzn.to/34SBuXf)	× 1	Amazon (https://amzn.to/34SBu

Software

Arduino IDE (https://www.arduino.cc/en/Main/Software)

How does an ultrasonic sensor work?

An ultrasonic distance sensor works by sending out ultrasound waves. These ultrasound waves get reflected by an object and the ultrasonic sensor detects them. By measuring how much time passed between sending and receiving the sound waves, you can calculate the distance between the sensor and the object.

Distance (cm) = Speed of sound (cm/µs) × Time (µs) / 2

Where **Time** is the time between sending and receiving the sound waves in microseconds.

Ultrasonic distance sensors working principle. Source: https://www.maxbotix.com/ (https://www.maxbotix.com/)

Note that you need to divide the result by two. This is because the sound waves traveled from the sensor to the object and back from the object to the sensor. So the distance between the sensor and the object is only half the distance that the sound waves traveled.

For more information on how ultrasonic sensors work, you can check out my article on the HC-SR04 (https://www.makerguides.com/hc-sr04-arduino-tutorial/). In this article, the working principles of an ultrasonic distance sensor are explained in greater detail.

Information about the sensor

The MB1240 XL-MaxSonar-EZ4 (https://amzn.to/2ZNESA7) is an ultrasonic distance sensor made by MaxBotix Inc. MaxBotix is a US based manufacturer that specializes in ultrasonic sensors. They make sensors for all kinds of applications, both for indoor and outdoor use.

The MB1240 is one of their most popular sensors and is ideal for robotic applications. The XL-MaxSonar-EZ family has a high tolerance for interference from acoustic or electric noise. This means you can use it for robotic applications with multiple motors and servos. The MB1240 that I used in this tutorial has a range of up to 765 cm and offers a 1 cm resolution. Maxbotix also sells sensors with a 1 mm resolution (HRLV-MaxSonar-EZ (https://amzn.to/2ZMseRG)) and higher refresh rate (LV-MaxSonar-EZ (https://amzn.to/2ZJR6cY)). The HR line of sensors also features internal temperature compensation.

TM1240 XL-MaxSonar-EZ4 Specifications

Operating voltage	3.3 – 5.5 V
Operating current	3.4 mA average (100 mA peak)
Range	(0)* 20 – 765 cm
Resolution	1 cm
Frequency	42 kHz
Reading rate	10 Hz
Sensor outputs	Analog voltage, pulse width, RS232
Overall dimensions	22.1 x 19.9 x 25.11 mm
Operating temperature	0 – 65 °C
Advantage	Small, light weight, narrow beam, automatic calibration (voltage, humidity, ambient noise), firmware filtering, easy to use
Made in	USA
Cost	Check price (https://amzn.to/2HKkb1P)

*The sensor has virtually no dead zone, objects closer than 20 cm range as 20 cm.

For more information, you can check out the datasheet here:

MB1240 Datasheet

MaxBotix sensor outputs

As you might have seen in the specifications table above, the MaxBotix sensors of the MaxSonar family have different outputs: analog voltage, pulse width and RS232 serial (I2C sensors are also available (https://amzn.to/32lDv7F)). In this tutorial we will look at both the analog voltage and pulse width outputs.

Analog voltage

This is probably the easiest way to read the measured distance from the sensor. The analog voltage output of the sensor outputs a linear voltage that gets larger as a target moves further away from the sensor.

Analog voltage output. Source: https://www.maxbotix.com/

(https://www.maxbotix.com/)

We can read this output with a microcontroller like the Arduino and calculate the distance by multiplying the reading by a constant scaling factor (this factor depends on the exact sensor type, see datasheet).

Pulse width

Another option is to use the pulse width output. This pin outputs a pulse width representation of the distance. You can use the `pulseIn()` function in the Arduino code to read the length of this output pulse in microseconds (μs). To get the distance, you need to multiply this reading with a constant scaling factor. For the MB1240 (XL-MaxSonar sensors), the scaling factor is 58 μs/cm. So you can simply divide the TOF reading by 58 to get the distance in centimeters.

Pulse width output. Source: https://www.maxbotix.com/ (https://www.maxbotix.com/)

For other types of sensors, you can find the scaling factors in the datasheets.

MaxBotix MB1240 vs HC-SR04

When shopping for an ultrasonic distance sensor, you have probably come across the popular HC-SR04 as well. This low-cost sensor is available from many different Chinese manufacturers. The HC-SR04 and the MB1240 operate on a similar principle, but there are some key differences (mostly in quality and price).

The first thing you might notice is the difference in size. The MaxBotix MB1240 only uses one ultrasonic transducer for both sending and receiving the sound waves. The HC-SR04, on the other hand, uses two. This means that the MB1240 is considerably smaller.

Mounting an HC-SR04 sensor can be difficult and often requires tiny screws, whereas the MB1240 has two holes for M3 bolts.

For me, the major advantages of the MaxBotix sensors are the automatic real-time calibration, small beam angle, high noise tolerance, and onboard filtering.

The MB1240 has a really narrow beam (https://www.makerguides.com/wp-content/uploads/2019/09/Beam-Pattern-MaxBotix-MB1240-XL-MaxSonar-EZ4-Ultrasonic-Distance-Sensor.jpg) across its entire measuring range (for large objects). This makes it ideal for accurately mapping a room (obstacle avoiding robots) and means you won't get early reflections from objects close to the sensor. The HC-SR04 has a beam in the shape of a 15° 3D cone. This means that you can't accurately measure objects that are far away from the sensor. The beam is simply too wide and will pick up objects that are closer to the sensor.

Furthermore, I like the continuous operation of the sensors (more on this later) and the analog output. The analog output makes programming the sensors super easy. These extra features are all very nice but note that the price is also quite a bit higher than that of the Chinese sensors (https://amzn.to/2HKkb1P). All in all, if you are looking for a high-quality reliable sensor then the MaxBotix sensors could be a good option. For a comparison, check out the table below.

MB1240 vs HC-SR04 comparison table

	MB1240	HC-SR04
Operating voltage	3.3 – 5.5 V	5 V
Operating current	3.4 mA average (100 mA peak)	15 mA
Range	(0) 20 – 765 cm	2 – 400 cm
Resolution	1 cm	>3 mm
Beam pattern	see here (https://www.makerguides.com/wp-content/uploads/2019/09/Beam-Pattern-MaxBotix-MB1240-XL-MaxSonar-EZ4-Ultrasonic-Distance-Sensor.jpg)	15° cone
Frequency	42 kHz	40 kHz
Sensor outputs	Analog voltage, pulse width, RS232	Pulse Width
Dimensions	22.1 x 19.9 x 25.11 mm	45 x 20 x 15 mm

	MB1240	HC-SR04
Temperature compensation*	No	No
Filtering	Yes	No
Real time auto-calibration	Yes	No
Noise tolerance	High	Low
Made in	USA	China
Price	Check price (https://amzn.to/2HKkb1P)	Check price (https://amzn.to/2MWkIlk)

*What you might not know, is that the speed of sound strongly depends on the temperature and humidity of the air. The speed of sound in air increases about 0.6 meters per second, per degree centigrade.

Both the HC-SR04 and the MB1240 do not compensate for a change in air temperature during operation. The XL-MaxSonar and LV-MaxSonar sensors assume an air temperature of 22.5 degrees Celsius. The HR line of sensors (https://amzn.to/2ZJF9I8) features internal temperature calibration, so you don't need to add any sensors yourself. If you would like to see an example that includes a temperature sensor to calibrate the speed of sound in real-time, take a look at this article (https://www.makerguides.com/hc-sr04-arduino-tutorial/#example-code-hc-sr04-with-dht11-temperature-sensor-and-arduino).

Wiring – Connecting MaxBotix MB1240 to Arduino UNO

As mentioned in the introduction, MaxBotix sensors can be operated in different modes. The wiring diagrams below show you how you can connect the MB1240 sensor to the Arduino for analog voltage or pulse width operation.

You can solder the wires directly to the sensor, or install some header pins (that is what I did) or a connector.

Analog voltage wiring diagram

The connections are also given in the following table:

MB1240 Connections – Analog voltage

MaxBotix MB1240 Sensor	Arduino
GND	GND
V+	5 V
Pin 3	A0

Pulse width wiring diagram

MB1240 Connections – Pulse width

MaxBotix MB1240 Sensor	Arduino
GND	GND
V+	5 V
Pin 2	Pin 2

MaxBotix MB1240 Arduino example code – Analog voltage

With the following example, you can read out the measured distance and display it to the serial monitor. As you can see, the code is very simple but you can find some explanation on how it works below.

You can upload the example code with the Arduino IDE (https://www.arduino.cc/en/main/software).

```
/* Arduino example code for MaxBotix MB1240 XL-MaxSonar-EZ4 ultrasonic distance sensor: analog voltage output. More info: www.makerguides.com */

#define sensorPin A0

int distance = 0;

void setup() {
  Serial.begin(9600);
}

void read_sensor() {
  distance = analogRead(sensorPin) * 1;
}

void print_data() {
  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" cm");
}

void loop() {
  read_sensor();
  print_data();
  delay(1000);
}
```

You should see the following output on the Serial Monitor (Ctrl + Shift + M).

MB1240 Serial Monitor output

How the code works

The first step is to define the connection pin. The statement `#define` is used to give a name to a constant value. When the program is compiled, the compiler will replace any references to this constant with the defined value. So everywhere you mention `sensorPin`, the compiler will replace it with A0 when the program is compiled.

```
#define sensorPin A0
```

Next, we need to create a variable to store the measured distance.

```
int distance = 0;
```

In the setup, we initialize serial communication at a baud rate of 9600. Later we will display the measured distance in the serial monitor, which can be accessed with Ctrl + Shift + M or Tools > Serial Monitor. Make sure the baud rate is also set to 9600 in the serial monitor.

```
void setup() {
  Serial.begin(9600);
}
```

After this, I created two functions: `read_sensor` and `print_data` .

In the read_sensor function, we simply read the analog voltage sensor output with the function `analogRead(pin)` . The Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map the input voltage between 0 and the operating voltage into integer values between 0 and 1023. On an Arduino Uno, this results in 5 volts / 1024 units or, 4.9 mV per unit.

The MB1240 uses a scaling factor of (Vcc/1024) per cm or 4.9 mV/cm when using a 5 V power supply. This makes converting the analogRead value to centimeters super easy, you can simply multiply the result by 1 (analogRead(sensorPin) = distance in centimeters).

```
void read_sensor() {
  distance = analogRead(sensorPin) * 1;
}
```

In the print_data function, we print the measured distance to the serial monitor.

```
void print_data() {
  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

In the loop, we first call the read_sensor function to get the distance and then call the print_data function to send it to the serial monitor. I added a delay of 1000 milliseconds, but you could reduce this to 100 if you want. The reading frequency of the MB1240 is 10 Hz so you can take 10 readings per second.

```
void loop() {
  read_sensor();
  print_data();
  delay(1000);
}
```

MaxBotix MB1240 Arduino example code – Pulse width

In this example, we will be using the other output of the sensor: the pulse width output.

```
/* Arduino example code for MaxBotix MB1240 XL-MaxSonar-EZ4 ultrasonic distance sensor: pulse width output. More info: www.makerguides.com */

#define sensorPin 2

long distance = 0;
long duration = 0;

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

void read_sensor() {
  duration = pulseIn(sensorPin, HIGH);
  distance = duration / 58;
}

void print_data() {
  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" cm");
}

void loop() {
  read_sensor();
  print_data();
  delay(1000);
}
```

Code explanation

After defining the connection pin, I created two variables: `duration` and `distance`. Duration stores the length of the pulse sent by the sensor. The distance variable is used to store the calculated distance.

```
long distance = 0;
long duration = 0;
```

In the setup, besides initializing serial communication, we also need to set the sensorPin as an INPUT. For this we use the function `pinMode(pin, mode)`.

```
void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
}
```

The read_sensor function is different from the previous example. Now we will not measure the analog voltage output, but the length of the pulse sent by the sensor. For this we use the function `pulseIn(pin, value)`. This function waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. It returns the length of the pulse in microseconds.

After this, we can calculate the distance in centimeters by dividing the duration measurement by 58. For other MaxBotix sensors, you can find this scaling factor in the datasheet.

```
void read_sensor() {
  duration = pulseIn(sensorPin, HIGH);
  distance = duration / 58;
}
```

The rest of the code is the same as the previous example.

Trigger mode operation

All the MaxSonar sensors will operate in free-running mode by default. What this means is that the sensor will continue to range until the power is removed from the sensor. It sends out twenty 42kHz waves every 99 ms (10 Hz reading rate for MB1240, see datasheet for other sensors).

This is generally the easiest way to operate the sensor, as you do not have to trigger it yourself and can just take an analog voltage or pulse width reading to get the distance.

For some applications, like when running the sensor from a battery, it can be better to operate the sensor with a trigger. What this means is that you can tell the sensor to start a ranging cycle, but only when it is instructed to. This way, you can control the highest current draw from the sensor which is when it transmits a sonar pulse.

The sensor draws the most current when it sends out a sonar pulse.

To operate the sensor with a trigger, we will use an extra connection between pin 4 of the sensor and the Arduino. When you do not connect anything to this pin, like in the previous examples, the sensor will range at the refresh rate of the sensor mentioned in the sensor's datasheet.

To trigger the sensor when needed, you need to connect pin 4 to a logic low. When you want to take a reading, you have to pull pin 4 high for a minimum of 20 µs. The sensor will then start a ranging cycle.

The wiring diagram below shows you which connections you need to make for this example.

MaxBotix MB1240 with momentary push button and Arduino wiring diagram

In this example, we will be using a momentary push button to trigger the sensor. You can buy these nice round push buttons on Amazon (https://amzn.to/2QhXhWi), that you can just plug into a breadboard. Connect one of the legs to ground and the diagonally opposite leg to Arduino pin 4.

The connections are also given in the table below.

MB1240 Connections – Trigger mode

Pin	Arduino
GND	GND
V+	5 V
Pin 2	Pin 2
Pin 4	Pin 3
Button pin 1	Pin 4
Button pin 2	GND

MaxBotix MB1240 Arduino example code – Trigger with push button

You can use this example sketch to control the sensor with a trigger. In this case, the sensor will take a reading when you press the button and display the distance measurement on the Serial Monitor. You can also just call the read_sensor function when you want to take a reading.

```
/* Arduino example code for MaxBotix MB1240 XL-MaxSonar-EZ4 ultrasonic distance sensor with push button. More info: www.makerguides.com */

#define readPin 2
#define triggerPin 3
#define buttonPin 4

long distance = 0;
long duration = 0;

int buttonState = HIGH;
int previous = HIGH;
long time = 0;
long debounce = 200;

```

```
void setup() {
  pinMode(readPin, INPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  Serial.begin(9600);
  delay(3000);
  Serial.println("Sensor is ready, waiting for button press!");
}

void read_sensor() {
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(20);
  digitalWrite(triggerPin, LOW);
  duration = pulseIn(readPin, HIGH);
  distance = duration / 58;
  delay(100);
}

void print_data() {
  Serial.print("distance = ");
  Serial.print(distance);
  Serial.println(" cm");
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == LOW && previous == HIGH && millis() - time > debounce) {
    read_sensor();
    print_data();
    time = millis();
  }

  previous = buttonState;
}
```

You should see the following output in the Serial Monitor (Ctrl + Shift + M).

How the code works

The first step is to define the connections. We will be using the pulse width output of the sensor to read the distance.

```
#define readPin 2
#define triggerPin 3
#define buttonPin 4
```

Besides the duration and distance variables that we used in the previous example, we also need some new variables to store the state of the button. The time and debounce variables are used to debounce (https://www.arduino.cc/en/Tutorial/Debounce) the input.

You can increase the debounce time if you are getting false triggers.

```
long distance = 0;
long duration = 0;

int buttonState = HIGH;
int previous = HIGH;
long time = 0;
long debounce = 200;
```

In the setup, we set the triggerPin as output and the read and buttonPin as input. Note that I used INPUT_PULLUP in the pinMode function. There are 20K pullup resistors built into the Atmega chip that can be accessed from software. This setting pulls the buttonPin HIGH when it is not pressed and it will turn LOW when you press the button.

Next, we set the triggerPin LOW, so the sensor will not start ranging.

To print the sensor data, we begin serial communication at a baud rate of 9600.

```
void setup() {
  pinMode(readPin, INPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  Serial.begin(9600);
  delay(3000);
  Serial.println("Sensor is ready, waiting for button press!");
}
```

After this, I defined two functions, the read_sensor and print_data function.

In the read_sensor function, you can see that we set the triggerPin high for 20 microseconds. This will tell the sensor to send out a sonar pulse. Next, we read the length of the output pulse and convert it into the distance (this is the same as the previous example). I added a delay of 100 ms, as this is the minimum time between readings.

The print_data function is the same as in the previous examples.

```
void read_sensor() {
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(20);
  digitalWrite(triggerPin, LOW);
  duration = pulseIn(readPin, HIGH);
  distance = duration / 58;
  delay(100);
}
```

In the loop, we first read the state of the button (pressed / not pressed) and store it as buttonState. The next line checks if you have pressed the button (i.e. the input went from HIGH to LOW) and if you have waited long enough since the last press to ignore any noise.

If this is true, it calls the read_sensor and print_data function and resets the timer.

```
void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == LOW && previous == HIGH && millis() - time > debounce) {
    read_sensor();
    print_data();
    time = millis();
  }

  previous = buttonState;
}
```

Finally, the previous variable is set to the current buttonState.

CAD

MaxBotix provides free CAD files for all their sensors on their website. This makes designing custom parts or brackets to use with the sensor super easy. You can download a zip file with a 3D model of the sensor below (7 different file formats). More models of different sensors can be found on their website (https://www.maxbotix.com/Ultrasonic_Sensors/MB1240.htm).

XL-MaxSonar-EZ.zip

Conclusion

In this article, I have shown you how you can use the MaxBotix MB1240 XL-MaxSonar-EZ4 ultrasonic distance sensor with Arduino. I hope you found it useful and informative. If you did, please **share it with a friend** who also likes electronics and making things!

I would love to know what projects you plan on building (or have already built) with this sensor. If you have any questions, suggestions or if you think that things are missing in this tutorial, **please leave a comment down below**.

Note that comments are held for moderation to prevent spam.

Beginner

4
SHARES

What to read next?

LM35 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/lm35-arduino-tutorial/)

TMP36 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/tmp36-arduino-tutorial/)

Arduino Nano Board Guide (Pinout, Specifications, Comparison) (https://www.makerguides.com/arduino-nano/)

The complete guide for DS18B20 digital temperature sensors with Arduino (https://www.makerguides.com/ds18b20-arduino-tutorial/)

How to use an IR receiver and remote with Arduino (https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/)

Trackbacks

HC-SR04 Ultrasonic Sensor Arduino Tutorial (5 Examples) (https://www.makerguides.com/hc-sr04-arduino-tutorial/) says:
September 26, 2019 at 6:04 pm (https://www.makerguides.com/maxbotix-mb1240-arduino-tutorial/#comment-657)
[…] MaxBotix MB1240 ultrasonic distance sensor Arduino tutorial […]

(https://www.makerguides.com/arduino-motor-shield-stepper-motor-tutorial/)

How to control a Stepper Motor with Arduino Motor Shield Rev3 (https://www.makerguides.com/arduino-motor-shield-stepper-motor-tutorial/)

(https://www.makerguides.com/sharp-gp2y0a710k0f-ir-distance-sensor-arduino-tutorial/)

How to use a SHARP GP2Y0A710K0F IR Distance Sensor with Arduino (https://www.makerguides.com/sharp-gp2y0a710k0f-ir-distance-sensor-arduino-tutorial/)

(https://www.makerguides.com/am2320-arduino-tutorial/)

AM2320 digital temperature and humidity sensor Arduino tutorial (https://www.makerguides.com/am2320-arduino-tutorial/)