

## Station 2 – Park Distance Control System

### Attention, obstacle!

Those of you, who will get their driver's license soon, will have to deal with one huge nightmare: parking in reverse! You move your car back and forth, but in the end, you are still not in the right spot, and you do not take your eyes off the mirrors for a second, because you fear, that you might bump into the car behind you.

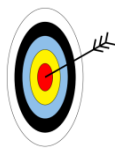


[1]

Thank goodness, that most modern cars come with an integrated park distance control system, which keeps you from bumping into other cars by the help of loud, increasingly fast beeping.

But, how does such a park distance control system work? How does the car know the distance? And how does distance become a warning signal?

The answer is hidden: "invisible light"! Or rather: infrared light, which you are familiar with from infrared remote controls. Light in the infrared spectrum is invisible to the human eye. It is sent by a sensor, reflected on a surface, and then registered by a receiver.



In this worksheet, you will learn,...

- × how to measure distance by the help of infrared light.
- × how to create warning signals from the distance.
- × how to realize this with an Arduino.

Doing that, you will hopefully prevent any bumps in the buffer bar of the test vehicle. ☺

### Necessary Components:

The circuit, which you have to install to realize an infrared park distance control system, is not complicated. Besides the Arduino, you need the following components:

- 2 wires (red and blue)
- a Piezo signaler (figure 2)
- an infrared distance measuring sensor (figure 3), which will just be called IR sensor in the following



[2]



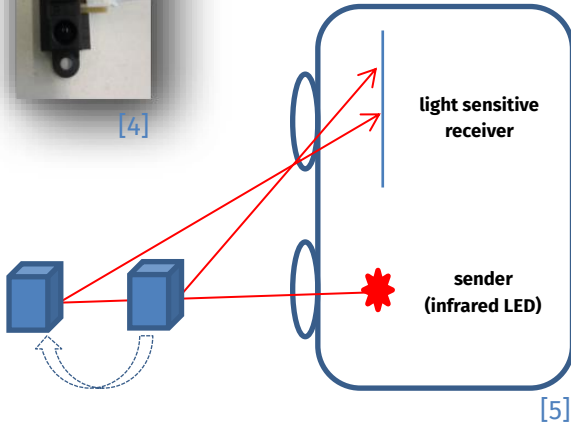
[3]

## Station 2 – Park Distance Control System

### The IR Distance Measuring Sensor



In this workshop, you use the *Sharp GP2Y0A21YK*, which is a complex device with an integrated connection, that can measure distances between 10 and 80 centimeters pretty accurately.



To measure distance, the sensor uses two eyes: a sender and a receiver. The sender sends an infrared light, which is reflected by a surface, and then hits the receiver – always in a different angle depending on the distance (see figure 5). With this information the device can calculate the distance using a particular mathematical method.

### The Piezo Signaler

The acoustic Piezo signaler is a so called “buzzer” or “beeper”. It is a small device, which is electronically controlled, and produces a certain sound. Thus, the Piezo is an acoustic output. It is used wherever loud sounds need to be generated quickly for warning or notification, for example in smoke detectors, in microwaves or as beeping in a park distance control system.

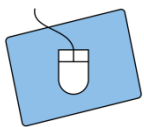


### The Analog Pins

Up to now, you only know digital pins, which are used to connect e.g. an LED. At these pins, only binary values can be read in and out – on/off, 0/1 or the known values high and low for a high or low voltage. An IR sensor, for example, can not only register two values, but measures many intermediate values in the range between the minimum and maximum value. On the Arduino, this is realized by analog pins (A0 to A5), where a whole range can be measured.

## The Construction of the Circuit

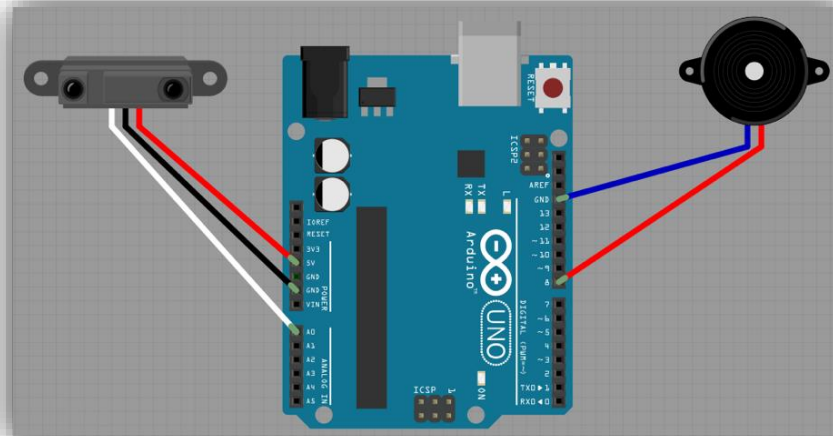
**Hint:** That time, you do not need the breadboards.



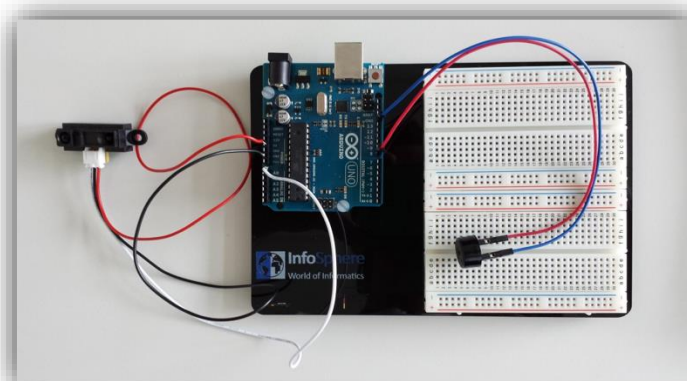
1. Use a **blue** wire to connect the **minus** pole (the shorter leg) of the Piezo with the GND pin of the Arduino.
2. Use a **red** wire to connect the **plus** pole of the **Piezo** with one of the **digital pins**, which you will use to control the output.  
**Hint:** Do not use pin 0 and pin 1.
3. The **IR sensor** is directly connected to the Arduino. Use the **black** wire to connect it to **GND** (near the analog pins), the **red** cable to connect it to **5V** and the **white** cable to connect it to one of the **analog pins**, which you will use as the input pin to measure the distance.

Station 2 – Park Distance Control System

**Hint:** The following figures show a sample solution. Your solution may look different (e.g. the pins, which you have chosen), because there is no unique solution.



[7]



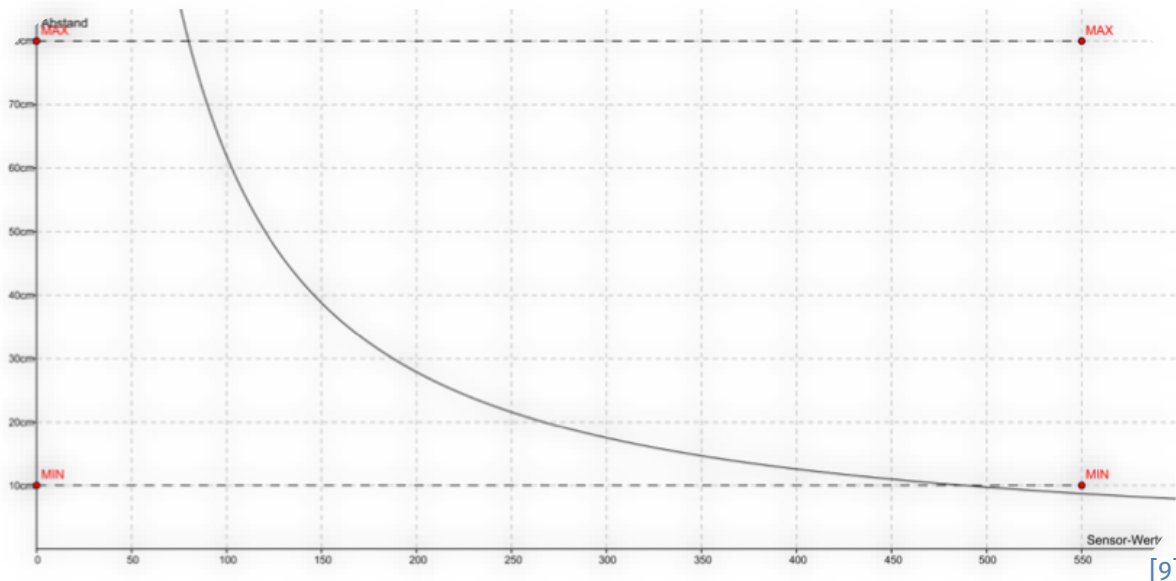
[8]

## Station 2 – Park Distance Control System

### Manuel Distance Measurement

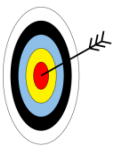
First of all, you will read in the data, which the IR sensor measures with the help of a little program, and then display them in the serial monitor. So, your first task is to use the IR sensor to measure the distance, and to assign these values to the different centimeter values in the table. This is an important step, because you need to know how the measured values match with the centimeter values, which are required for the distance.

The following figure shows the output values of the IR sensor for distances between 10 and 80 centimeters.



The analog input pins of the Arduino will always give you a value between **0 and 1023**. Thus, you need to be able to translate these values into centimeters. However, this is not that simple, there is no quick mathematical formula to do this conversion.

**Hint:** Later, you will get a formula, which you can use to compare the data, that you have filled in.



In this step, you will learn to...

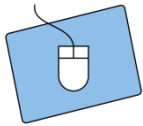
- × read in the sensor values using an analog pin, and to
- × display these values on the serial monitor.



1. Open a new sketch, and save it under a meaningful name.
2. Implement the default structure, which you know from the introduction.
3. Before you can start working with `setup()` and `loop()`, you have to declare (just as in the introduction) an **int variable** for the analog input pin. Assign the correct pin number.

**Hint:** Even if the analog pins are labeled A0, A1 etc., you only use the numerical value for the assignment in the sketch!

## Station 2 – Park Distance Control System



4. Now, you have to declare a second variable. In this variable you will store the data of the IR sensor, so that you can use it whenever you need. Do not forget to give it a meaningful name.
5. Let's do the `setup()`: How can you start the transmission to the serial monitor?
6. Now, you do the `loop()`:
  - a. Here, the most important step is the reading of the sensor values. Different from setting the voltage at the digital pin to high or low, you need to use the following expression to read in the sensor values:  
`sensorDataVariable = analogRead(sensorPinVariable);`  
 So, assign the values to the variable of step 4 by using "=", and chose the correct pin.  
`_____ = _____( _____ );`
  - b. Now, you need the output on the serial monitor, which is again introduced with `Serial`. Use inverted commas to include a short description of the output.
  - c. Add a second output, which should include the variable of the sensor values. This time, you do not use inverted commas, which are only used for text. As you want to display sensor values, which are stored in a variable, you need brackets.
  - d. To prevent the values from being displayed too fast on the serial monitor, use a `delay()` after the output commands so that the values are only updated every one or two seconds.
7. Test your sketch. Connect the Arduino to your computer, run the program, and observe the sensor values.

Soon, you will notice that the distance values are not in the range of 10 to 80 centimeters. What is the range of the values?

Minimum = \_\_\_\_\_ Maximum = \_\_\_\_\_

It is not enough to know the minimum and maximum values, if you want to get a meaningful classification of distances. Thus, precision work is required. Use a long strip of paper and a folding rule to find out what sensor values are output at 10, 20, 30 cm, and so on. Note these values in the table.

**Hint:** The IR sensor measures most accurately, if you use a white piece of paper to reflect the infrared light.

Distance	10 cm	20 cm	30 cm	40 cm	50 cm	60 cm	70 cm	80 cm
Sensor Values								



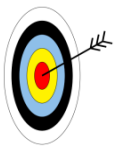
## Station 2 – Park Distance Control System



Do not get frustrated! Even if you hold the sensor steady, it still might give you different values for one and the same distance. Just decide on a reasonable average value.

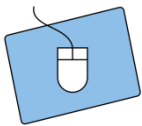
### A Simple Warning Signal

Let's get back to programming. For now, the IR sensor measures the distances, but you do not have a warning signal yet. Therefore, you have to use the second device, the **Piezo signaler**.



In the next step, you will learn to...

- × include the Piezo into your sketch.
- × generate an acoustic signal depending on a sensor value.



1. Save your sketch under a different name (Save under). From now on, you will expand, change and adjust your program again and again.
2. Declare a new int variable for the Piezo, and assign it the digital output pin, which you used to connect the Piezo to the Arduino.
3. In `setup()`, you have to define the Piezo as output (since it creates a sound); this works the same way as with the LED of the introduction.  
`pinMode( _____, _____ );`
4. In `loop()`, you need the good, old `if else` construction, which you also know from the introduction.

### The If Else Construction

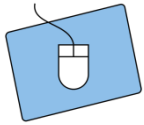
If you want to define in your sketch, that something should only happen under a certain condition – for instance to create a sound, if the distance is too small –, you can use the so called **if else construction**.

```
if(condition){
  //expression, if condition is true
}
else {
  //expression if condition is false
}
```

Often, the `condition` is a mathematical comparison (`>`, `<`, `>=`, `<=`, `==`). There is also the **logical operator &&** (logical and) to check whether two conditions are true at the same time:

```
if(condition1 && condition2) {
  //expressions if both conditions are true
}
```

## Station 2 – Park Distance Control System



5. Now, you have to use an `if else` construction. To do this, implement the basic structure as described above.
6. You want the Piezo to beep (create a continuous sound), if the distance to the IR sensor is too small (here: under 40 cm). Which variable do you need to create this condition? Have a look at the table: What limit do you have to compare with?
7. If your condition is true (that means: if the distance is too small), the Piezo should create a continuous sound. For that, you have to set the **digital output pin to high** (just like you did for the LED).
8. In any other case, there should not be any sound. Think about which command you can use in your `if else` construction to turn the piezo off again. Use the following scheme to note your considerations:

```
if( _____ ) {
    _____;
} else {
    _____;
}
```

9. Done! Load the sketch on the Arduino, and test it. Are you happy with the distance, which you have set? Sounds the tone too early or too late? Adjust, if necessary.

### What Now?

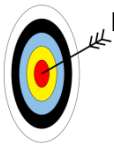
Do you have some other ideas or suggestions for improvements? Discuss them in your group, and note them! In the next two sections, you will keep on improving your park distance control system.

## Station 2 – Park Distance Control System

### Beep, Beep!

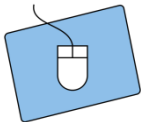
Or not, because: So far, your park distance control system makes one **continuous sound**, if the distance is too small. But a driver **wants to know how much space is left**, i.e. whether there is still no danger, he is approaching the obstacle, or is already very close to it.

You can realize all this by using the Piezo, which beeps at different rates.



In this improvement, you will learn to...

- ✘ set up distance intervals using the sensor values.
- ✘ create a beep using `digitalWrite( pinName, conditon);` and `delay(time)`.



1. You are already done with the circuit. So, you can directly start programming. Save your old sketch under a new name.
2. You have to do some changes in the `loop()`, or rather in the `if else` statement:
  - a. Have a look at your table on page 5. Use the following table to create a suitable division of the sensor values, and the distance in centimeter. The following three ranges should be considered:
    - Range 1: great distance → no sound
    - Range 2: obstacle comes closer → beeping
    - Range 3: obstacle is quite close (danger) → beeping faster

	Range 1 (great distance)	Range 2 (obstacle comes closer)	Range 3 (danger)
Centimeter	>	-	<
Sensor Values	<	-	>

- b. Now, you have to adjust your `if else` statement by including the different ranges. You have to change your one `if else` statement (which can only distinguish between two intervals) to three `if` statements.

**Hint:** Use the **mathematical comparisons**, and especially the **logical and (&&)**, and change the **speed** of the beeping by the help of `delays`. On the following page you can find a scheme for the three `if` statements.

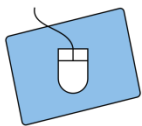


## Station 2 – Park Distance Control System

For each of the three ranges you need a separate `if` statement like this:

```

if(_____ ) {
  digitalWrite(_____, _____);
  delay(_____);
  digitalWrite(_____, _____);
}
if(_____ && _____) {
  digitalWrite(_____, _____);
  delay(_____);
  digitalWrite(_____, _____);
}
if(_____ ) {
  digitalWrite(_____, _____);
  delay(_____);
  digitalWrite(_____, _____);
}
    
```



Test your program. Load your sketch on the Arduino, and test your park distance control system. Adjust the chosen ranges or the `delays`, if necessary.

Done?! Congratulations! Thanks to your help, the test vehicle can now park in reverse very easily. 😊

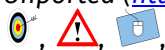


### List of references:

**Fig. 1** – Source: commons.wikimedia.org ([https://commons.wikimedia.org/wiki/File:Parking\\_Assist.jpg](https://commons.wikimedia.org/wiki/File:Parking_Assist.jpg)), Author: Nozilla, CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/deed.en>), 2022-06-27

**Fig. 2 to 6, 8, 9** – Source: InfoSphere

**Fig. 7** – Source: Screenshot of the Fritzing Software (<http://fritzing.org>), CC BY-SA 3.0 Attribution-ShareAlike 3.0 Unported (<https://creativecommons.org/licenses/by-sa/3.0/>), 2022-06-14

 – Quelle: InfoSphere