Informatics Enlightened

Station 1 – Sunflower

Efficient Sunbathing

For the survival of a sunflower, it is essential to gather as much sunlight as possible. That is the reason why sunflowers slowly turn towards the brightest spot in the sky.

Solar cells work on the same principle to turn as much sunlight as possible into electricity.

[1]

[2]

Using these worksheets, you will learn ...

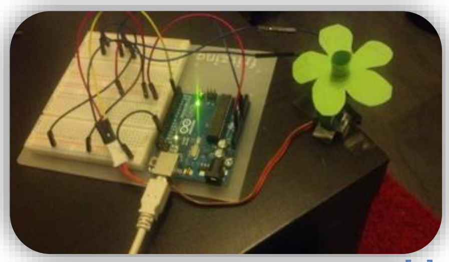
- to build your own sunflower.
- how to make it turn towards a source of light by combining the right electronic devices.

Required Components:

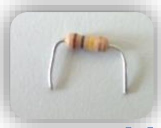
The final circuit of the sunflower on the Arduino breadboards only includes a handful of components. Besides the Arduino and the breadboards, you need the following items:

- a brightness sensor (figure 4)
- a 100 kΩ resistor (brown-black-yellow)
- a servo motor with a cross top (figure 6)
- 3 blue, 2 red, 2 yellow, 2 green wires
- a paper sunflower

In addition, you need a flashlight as a source of light to test your project. You can also use the flashlight of your smartphone.


[3]

[4]


[5]

[6]

[7]

[8]

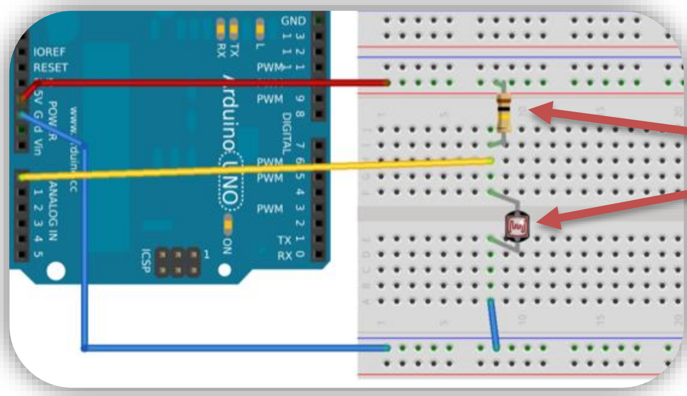
To make you turn your sunflower purposefully, every device you have to know about. Help you, these exercise sheets can.

Station 1 – Sunflower

Using the **brightness sensor**, the sunflower can recognize, if it is getting brighter or darker, and if it is moving to the right (towards the sun) or wrong (away from the sun) direction.

Brightness Sensor, or rather Light Resistor

The brightness sensor, which you are using, is a resistor. Depending on the incidence of light it lets more or less electricity through. The more light hits the sensor, the lesser the resistance: more electricity flows. To measure resistance, you have to examine the flow of electricity. A digital pin can only identify, if electricity flows (high) or not (low), but it cannot measure, how much electricity exactly flows. This is why you need the **analog pins**.


[9]

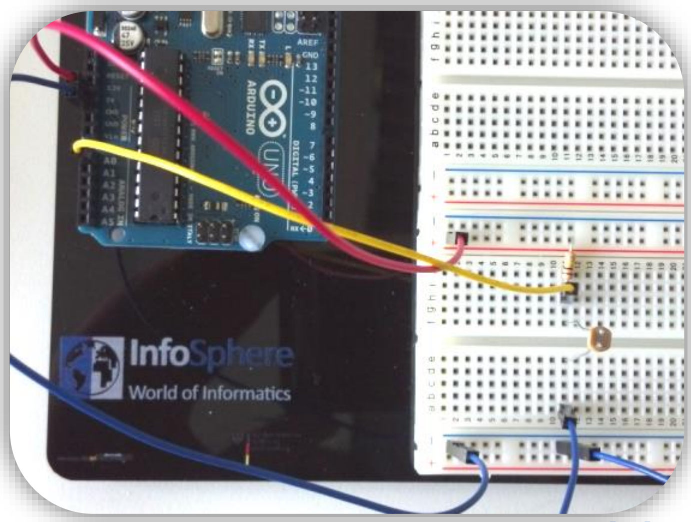
Required Components:

You need the following components to test your brightness sensor:

- a 100 kΩ resistor (brown-black-yellow)
- a brightness sensor
- 2 blue, 1 red, 1 yellow wire

Take a look at figure 9 and 10. There, you can see, what your circuit should look like.

The circuit is similar to the one, which you already built for the LED. In addition, you need a cable between the brightness sensor, and the resistor, which is connected to an analog pin.



[10]

Now, you can start programming, and working with the brightness sensor. The instructions on the next page explain, how to use the serial monitor to find out, which signals arrive at the analog pin.

Station 1 – Sunflower

1. Start a new sketch, and give it a meaningful name.
2. Create the default structure, which you know from the introduction.
3. First, you need to set up (in the **settings section**) all **variables**. Give them a meaningful name. You need:
   a. An `int` variable for the pin, which is connected to the **brightness sensor**. The value of the variable has to be the number of the analog pin.
      Hint: Even if the analog pins are labeled A0, A1, etc., you only use the numerical value for the assignment in the sketch.
      Name of the pin variable: ________________
   b. An `int` variable, which will save the number for the brightness, in this case 0.
      Name of the brightness variable: _______________
4. In `setup()`, you have to start the **serial monitor**. If you do not remember, how it works, feel free to look it up at station 0. Normally, analog pins are only used as input, therefore, no pin mode needs to be set, and the rest of `setup()` remains unchanged.
5. In `loop()`, you can now read the analog pin, and store the content in your brightness variable from 3.b.:
   ```
   brightnessvariable = analogRead(pinvariable);
   ```
6. You should output the variable for the brightness directly via the **serial monitor**.
7. The program will measure and output numbers on the screen faster than a human can read. So, to follow the serial monitor, you have to add a `delay`.
   Remember: 1 second = 1000 milliseconds.
8. Now, you can start testing. Probably, you will have to fix a few oversights. They happen to almost everyone, and are only noticed during testing. Start the program, and open the serial monitor. If the numbers are still output too fast, increase the delay value further.
9. Illuminate the head of the brightness sensor, and look at the output values on the serial monitor. Shield the sensor with your hand or a sheet of paper.
   Hint: In reality, it is never the same brightness. So, the values will always vary a little.
   Note your findings:

Dark:	high values	☐
	low values	☐
Bright:	high values	☐

Now, you know the range of the brightness sensor values, and you can tell, whether it is bright or dark. But the sunflower needs to know, whether it becomes brighter or darker. To achieve this, you can use another `int` variable, which stores the old brightness value, and an if else statement, which checks, whether the current brightness value is higher or lower than the old one.

1. Declare a new `int` **variable in the settings section**. This variable will later store the old brightness values. Give it a meaningful name, and the start value 0.
   Name of the variable: ____________________
2. Now, you need an **`if else`** statement, which compares the old to the current brightness value, and outputs via the serial monitor, whether it has become brighter or darker. For the condition, you can use the "greater than", and "less than" signs like in math class.

```
if( __________________ > _________________ ){
   Serial.println("darker");
}
else {
   Serial.println("brighter");
}
```

   Should it be placed before or after the light measurement? Before or after storing the brightness value (see 3.)?

   Tipp: Your program is processed from top to bottom. Consider, in which line the if else statement in `loop()` has to be inserted, so that the brightness values are compared correctly.
3. Now, store the measured light value in the new variable at the very end of the `loop()`:
   `oldBrightnessVariable = measuredValue;`
4. Test your program, and see, what happens on the serial monitor. If it does not work right away, do not worry. This happens regularly even to experienced programmers. Look for the error, and try to fix it.
5. Save your sketch. You will need it later for the flower.

Station 1 – Sunflower

Very good. You already know, how the brightness sensor works. You can detect brightness changes with the Arduino. Now, you can take care of the rotation.

Required Components:

- a servo motor with a cross top (you have to put the cross top on the motor)
- 1 blue, 1 red, 1 yellow wire

With the servo motor you can move your sunflower. Here, you will learn, how it works.

The Servo Motor

A servo motor is a motor, that can rotate to a certain angle. Your servo motor can rotate from 0° to 179°.

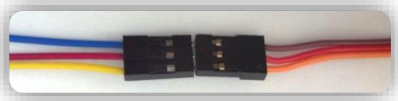
[6]

You can only tell the motor to rotate to a certain angle, not by a certain angle.

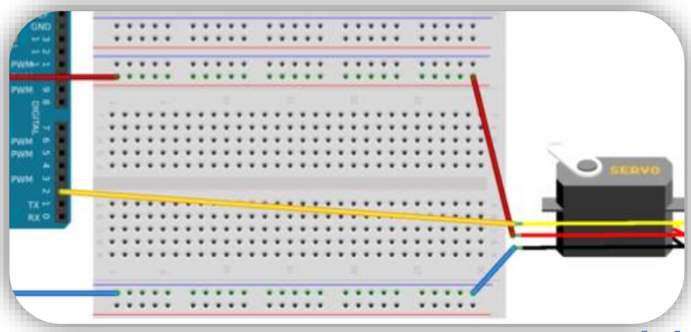
The Structure of the Circuit

You can keep your old circuit, because you will need it again for the sunflower.

1. Connect your wires with the servo motor.
   - brown → blue
   - red → red
   - orange → yellow

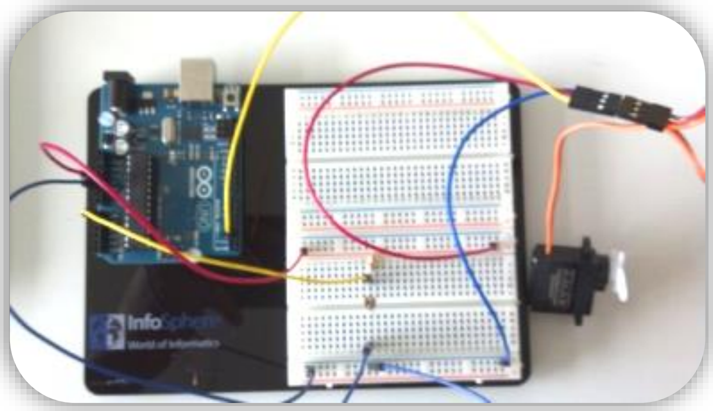

[11]

2. Connect the wires to the Arduino.
   - blue → minus bar
   - red → plus bar
   - yellow → digital pin



[12]

In figure 13, you can see, how the two circuits look like built side by side.


[13]

The Program

Now, you can start programming the Arduino to move the motor. For this, you will first learn, how to prepare the program, and then, how to move the motor step by step.

1. Open a new sketch, and save it under a reasonable name.
2. Implement the basic framework from the initial project, if it was not created automatically.
3. Write the following at the very top of your project:
   ```
   #include <Servo.h>
   ```
   This tells the Arduino software to make all servo functions available. This instruction must be in the very first line of the program with angle brackets, and semicolon.
4. Now, you can add the servo to your program. This works as with a variable. With that, you can later use all servo functions. Different from variables, servos do not get a start value:
   ```
   Servo servoName;
   ```
5. As with any component, that is controlled, you need an `int` variable to save the pin for the servo motor.
   Name of the servo pin: ___________________

Continue on the next page…

6. In **`setup()`**, you now have to tell the program, to which pin the servo is connected. To do so, insert the following:
   `servoName.attach(servoPin);`
7. In **`loop()`**, you can now move the motor. You can type in
   `servoName.write(degree);`
   and add a random number between 0 and 179 into the brackets.
   Hint: Do not add „°“.
8. Test your program, and look for mistakes, if it is not working.
9. Repeat step 7 and 8 with different numbers, and find out, where the motor can move to.
   Note your findings:
   When increasing the number of degrees, the motor rotates…

   clockwise. ☐

   counterclockwise. ☐

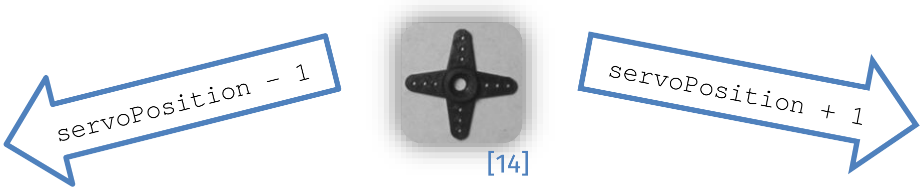
Great! You know, how to achieve, that the motor moves. Now, you can learn to use variables to make the motor turn slowly, and step by step from 0° to 179°.

1. Declare a new **`int`** variable, which will later store the new number of degrees (or position) for the motor. Give it a meaningful name. Assign the initial value 0 to it.
   Name of the variable for the servo position:_____________________________
2. In **`loop()`**, this position should always increase by 1. A variable can be increased by calling it up, adding 1, and saving the new value. That sounds more difficult than it really is:
   `servoPosition = servoPosition + 1;`
   Use this new position in your **`write`** statement, so that the motor can head for its new target. Thus, replace the number in your `write` statement by the name of your position variable.
3. A `loop` repetition happens so fast, that the motor would not have any time to move to the new position. Therefore, you should use a `delay` after `servoName.write().`
4. Test your program.
5. Change the `delay` value, and try again. Repeat that multiple times, and complete the following rule: If you increase the `delay` value, the motor gets…
   slower. ☐

   faster. ☐

You have learned to move the motor gradually in a certain direction. But probably, you will have noticed, that the motor stops at 179°. Now, you want the motor to move slowly back and forth between 0° and 179°. You can solve this with some if statements, and a new variable for the direction of movement.

1. Declare an `int` **variable,** which should save the direction. If the value is 1, the motor should turn clockwise. If the value is -1, the motor should turn counterclockwise. Assign the initial value 1 to it.



[14]

2. Use this variable for your position calculation by just replacing the +1 by the variable for the direction:

```
servoPosition = servoPosition + direction;
```

3. Since the start value is 1, nothing should have changed in the way your program works. The motor should still rotate from 0° to 179° now. Test it!
4. To make the motor change direction at the right moment, you need to insert two **if statements,** that set the direction variable to 1 or -1, if the motor would turn too far. In principle, it works as following:

```
if (new position == 179) {
set the direction variable to -1;
}
if (new position == 0) {
set the direction variable to 1;
}
```

5. Test your program. If you have done everything correctly, the motor should move back and forth between 0° and 179°.
6. Save the sketch! You will need it again for the sunflower.

Well done! Now, you know everything you need. You can finally build the circuit, and program your sunflower. If you want the orientation of the sunflower to be exact, you have to shield the brightness sensor, so that most of the light just comes from one direction. Otherwise, the brightness sensor will be too inaccurate.

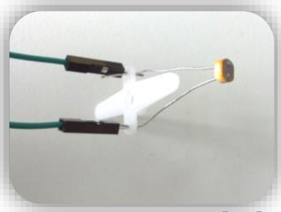
Required Components:

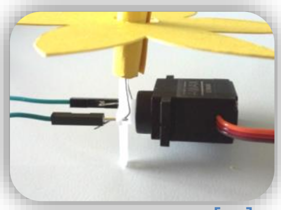
- 2 green wires
- 1 sunflower pattern

Attachment of the Sensor

1. Cut out the sunflower, roll up the rectangle, and put it through the hole in the center of the flower.
2. Take off the brightness sensor from the breadboard.
3. Attach it to the servo motor by pushing the wires through one of the holes at the opposite arms of the motor. Make sure, that **the wires do not touch each other**, otherwise, your circuit will not work.
4. Put now the green wires in the same holes. That way, the wires, and the sensor should stay in place on their own.
5. Then put the flower over the sensor, and one arm of the servo cross.

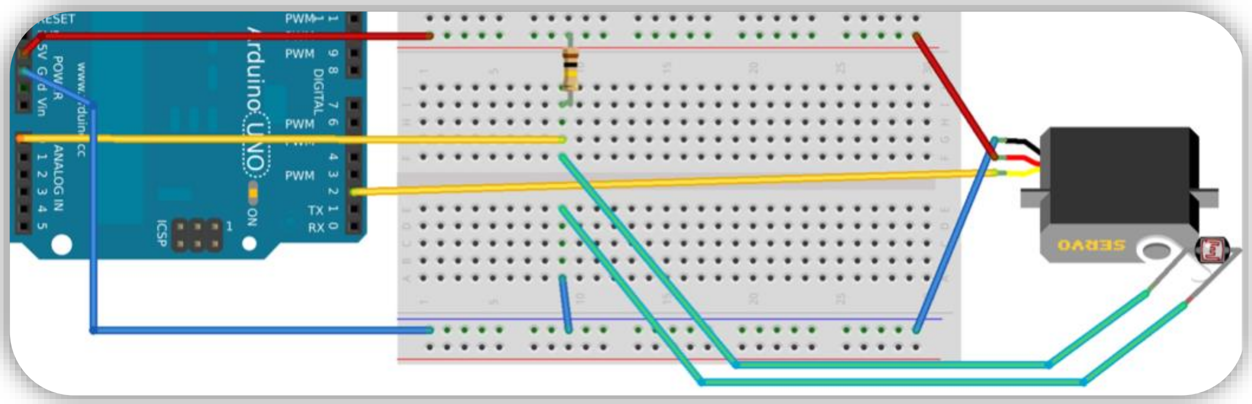
[15]


[16]


[17]

The Circuit

To connect the sunflower circuit, you have to plug in the green wires, where the brightness sensor was plugged in before.
Below, you can see an overview of the complete circuit.



[18]

Functionality of the Sunflower

Great, now you know everything to control the sunflower. The sunflower should always move in small pieces, and change its direction of movement, when it gets darker. Otherwise, it will keep the direction.

1. Open the second servo motor sketch, and use the „save under“ function to save it under a meaningful name. That way, you do not have to start all over again. Instead, you can expand your existing servo motor program. Your sunflower is almost done.
2. Tell the motor to only change its direction, if it is getting darker. Otherwise, the direction should be maintained.
   a. Take the variables for the two brightness values (new and old) from the program for the brightness sensor, and adjust `setup()` accordingly.
   b. Use, what you learned about the brightness sensor to adjust `loop()`, so that your program works like this:

```
void loop(){
1.  Measure and save brightness value
2.  If it is getting darker, change direction
3.  If position 179 oder 0: change direction
4.  Calculate new position for servo
5.  Move servo
6.  Wait until it has turned completely
7.  Save brightness value in the variable for old brightness
    value
}
```

3. Test your program with the help of a **flashlight** (of your smartphone). Also, observe the brightness values on the **serial monitor**. If it does not work, it may be due to an oversight in your **if else statements**. It can also be due to too large or too small values of your `delays`. It is also possible, that somewhere a `delay` is missing or too much. If the lighting conditions are bad, it may help to let the motor rotate several degrees in each step.

Done! Great! You have constructed your own sunflower. You can either try to optimize it, or continue with another station.

<u>*List of references:*</u>

Fig. 1 – *Source: pxhere.com (https://pxhere.com/de/photo/1448455), Author: sandmann 8888, CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), 2022-06-22*

Fig. 2 – *Source: pxhere.com (https://pxhere.com/de/photo/546340), CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), 2022-06-22*

Fig. 3 to 7, 10, 11, 13 to 17 *– Source: InfoSphere*

Fig. 8 – *Source: pxhere.com (https://pxhere.com/de/photo/901512), CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), 2022-06-22*

Fig. 9, 12, 18 – *Source: Screenshots of the Fritzing Software (https://fritzing.org/), CC BY-SA 3.0 Attribution-ShareAlike 3.0 Unported (https://creativecommons.org/licenses/by-sa/3.0/), 2022-06-14*

🎯, ⚠, 🖱, 🏁 – *Source: InfoSphere*