ROBOTICS

PHYSICAL COMPUTING

Traffic lights with a pushbutton

Author: Ivan Novosel

Institution: V. gymnasium, Zagreb

Co-funded by the Erasmus+ Programme of the European Union

Introduction

In an earlier exercise we have built a simple streetlight with three LEDs and programmed it to have the correct order of switching through the phases of an actual streetlight.

This time we will make a more complicated version. We will **add lights for pedestrians** and those will be **activated only if a button is pressed**. Streetlight should work like this:

1. Vehicle lights should be on by default. If no action is taken these should just go through normal phases for vehicles (go, stop, wait, ready).
2. If the button is pressed at any time than the next time vehicle wait phase (the red light is on) starts the pedestrians turn starts.
3. After the pedestrian turn finishes the vehicle lights go to normal operation.

Concurrent execution

To solve this problem we need to be able to switch the lights and check if the button is pressed at the same time. As the button can be pressed at any time we need to check if it is pressed continuously.

This type of problem is very common – how to make a computer do two or more things at once. Arduino has only one processor (as do most embedded computers) so we have to “fake it”. In general, **concurrent execution** is how we call programming tricks that help us perform several tasks at once.

The idea we will explore today is how to **quickly switch between two tasks.** Even though tasks can take a long time (example: red light needs to be on for several seconds) single commands take a very short time to execute, so to humans it will look like it’s happening at the same time.

The way how we programmed the streetlights in the previous version is to use the `delay(ms)` function to wait for a certain period. That is a problem because the delay function blocks Arduino from doing anything else in that time.

We can’t use the delay() function if we are making a concurrent algorithm!

Blink without using delay()

Documentation for **Blink without delay** on arduino.cc, URL: https://bit.ly/47VhwwE

Function `millis()` gives the time period (in milliseconds) from when Arduino started working till now

We will first explore how we can measure (and wait for) time intervals without using the `delay(ms)` function. For this we will adapt the first program that we did so far – blinking LED.

TASK: Run the example and read the code!

The whole example is available in Arduino IDE under *File→Examples→ 02.Digital→BlinkWithoutDelay* .

Inside the loop function is the part of code that replaces the `delay(ms)`:

```
...
unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
...
```

The idea is to make a timer. We save the moment when it started in the `previousMillis` variable. Variable `currentMillis` has the current time as the name says. If the difference between those becomes larger the interval of time that we wanted to wait has passed. So if the `interval` variable is set to 1000 this is the same as using a `delay(1000)`.

Always use >=

* * *

Time doesn't change in whole milliseconds, but in periods it takes to execute a command

Traffic lights without using delay()

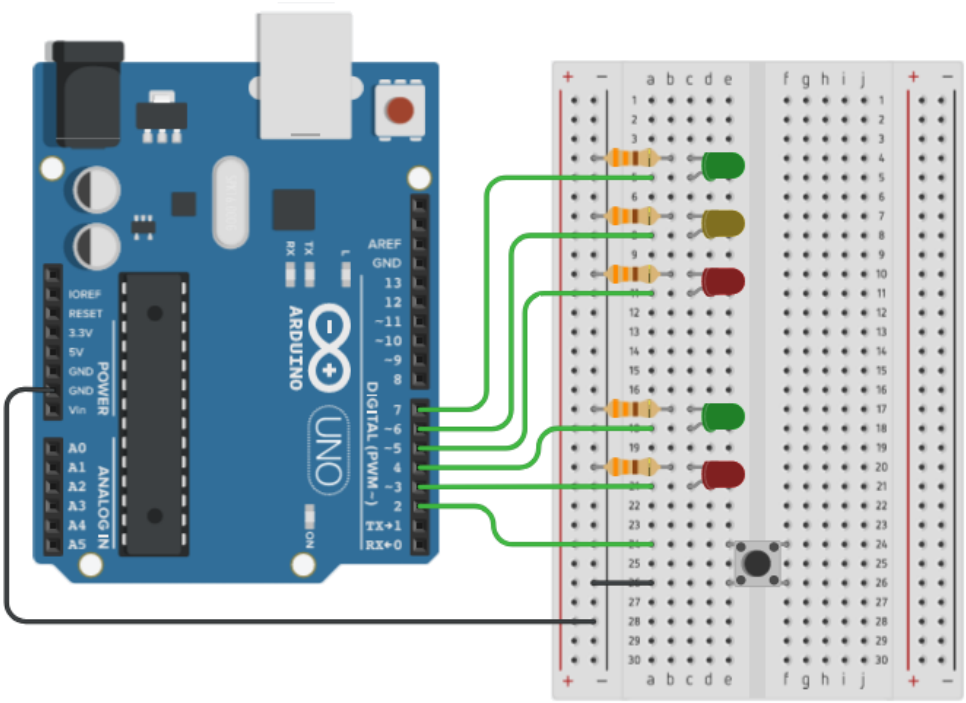
Now we have an idea of how to turn the lights on and off without using the delay function. Next step is to connect the LEDs for the streetlight and try to figure out how we can do that for several LEDs.

The challenge here is that the interval is changing as the next phase of the traffic comes. We need to change the interval for the timer and depending on the phase turn on different LEDs.

To do this we will use two new features of the C++ language that we haven't used before – **arrays** and the **switch-case** control structure.

Connecting the components

Connect all the necessary components, we will first make the vehicle part work as before, but without using delay, and then we will add the parts for pedestrians. That way there is less chance of getting lost in potential mistakes we make as we develop the program.



Picture 1 – the wiring of the components, three LEDs at the top are for vehicles, the two below that are for pedestrians as is the pushbutton.

Changing the state with arrays

More on **Arrays** on arduino.cc, URL: https://bit.ly/3OURHEk

Array is a type of data in C++ that enables us to make lists of connected information. For our needs:

```
const unsigned long vehicleIntervals[] = {3000,1000,3000,1000};
```

Here `vehicleIntervals` is the name of the array and the numbers in the `{}` brackets are lengths of the intervals. Each member of an array has an index number starting with 0. So we would get the first interval by using `vehicleIntervals[0]`. We will use these indexes to determine in which state is the traffic light:

- 0 – green light
- 1 – yellow light
- 2 – red light
- 3 – red+yellow light

We will keep the state we are in in a separate variable – the line `int vehicleState = 0;` just states we start with the green light.

Now we can change the if statement in the timer to this:

```
if (currentMillis- beginningMillis >= vehicleIntervals
[vehicleState]){ ...
```

The only thing that we changed here is the interval. We don't have a single number, but we select one from the array. Of course, we need to change these so we cycle through all the states, and depending on that we turn the LEDs on and off. We will use a **switch-case statement** for this.

```
if (currentMillis - beginningMillis >= vehicleIntervals[veh
icleState]){
    beginningMillis = currentMillis;
    switch (vehicleState){
        case 0:
        //things to do when the green light ends
        vehicleState = vehicleState + 1;
        break;
        case 1:
        //things to do when the yellow light ends
        vehicleState = vehicleState + 1;
        break;
        case 2:
        //things to do when the red light ends
        vehicleState = vehicleState + 1;
        break;
        case 3:
        //things to do when the red+yellow light ends
        vehicleState = 0;
        break;
      }
   }
```

When we start the switch-case we point it to the variable that will hold the different cases

If you omit the break; at the end of a case it will continue with other commands until it gets to a break or the end of swtich-case!

At the end of each case we progress to the next state.

The last one returns the state back to zero!

Switch-case checks if a certain variable has one of the desired values. In our case we look at `vehicleState`. Important thing is at the end of each case we need to change the `vehicleState` variable so need to the next index – that will change the interval as the index will point to the next member of the array.

TASK: Add the rest of the code for the LEDs

More on **switch-case statement** on arduino.cc, URL: https://bit.ly/44roxlH

HINT: if you get stuck this simulation has the whole example.

Simulation of **Traffic lights without delay** on Tinkercad, URL: https://bit.ly/3R5UkG5

Adding the pushbutton

Simulation of **Traffic lights with a pushbutton** on Tinkercad, URL: https://bit.ly/3OQPkCm

Now that we have the traffic lights for vehicles working, we can add the code needed for pedestrians. This has two parts, we need to check if the pushbutton was pressed, and correctly change the lights.

PHEW! This was a lot of work :) Congratulations on getting to the last step :)

As the pushbutton is connected directly to the Arduino we must use the INPUT_PULLUP mode, so we will read the button and save the state (LOW means the button was pressed): `if (digitalRead(pushPin)==LOW){ buttonPressed = true; }`

Now we can turn the LEDs for pedestrians – these intervals overlap with the ones for vehicles so we can just add it under right cases:

```
switch (vehicleState){
        case 0:
        //no changes here
        case 1:
        if (buttonPressed){
          // add the code for LEDs
        }
        // code for vehicles here
        break;
        case 2:
        if (buttonPressed){
          // add the code for LEDs
          buttonPressed = false;
        }
        // code for vehicles here
        break;
        case 3:
        //no changes here
      }
```

TASK: Add the rest of the code for the LEDs

Under case 1 the red lights for vehicles start so we also start the pedestrian lights!

Under case 2 the pedestrian lights need to stop but we also stop them from going on by themselves.

- **For bigger programs work in stages.**
- **No delay() for concurrent execution, use timers instead.**
- **Arrays are useful to save connected information.**
- **Switch-case is an alternative to if-else.**