Arduino For Beginners - Final Project

Time to start the final project of this course !

In this project you will practice with all the components and functionalities you've seen in the course. If you need a refresher, please go back to the corresponding section in the course.

The basics of the app is to check the distance in front of the circuit/robot, and lock the application when an obstacle is too close. By "lock" I mean that the behavior for the output (LEDs and LCD screen) will override the normal behavior. And to "unlock" the application, you'll have to manually press on a button. Please watch the intro video so it makes more sense.

On top of that, you will add several functionalities to increase the interactiveness of the app.

How to start the project

1. Make sure to watch the intro video showing you the result you'll have at the end.
2. Read the specs below to get more details.
3. Try to imagine what steps you'd need to take to finish the project.
4. Read the project step orders to get an idea of how to start and progress.
5. Start working on the project!
6. While doing the project, keep this document on the side and don't hesitate to come back to previous course lessons for a refresher. If you are stuck, watch the next step solution.

Specs per component

I have classified the functionalities per component to make it simpler. This is not necessarily the order of the solution steps.

- **Ultrasonic sensor:** read the distance every 60 ms. If distance is lower than 10cm, lock the application.
- **Push button:** when the app is locked, press + release the button to unlock the app.
- **Error LED (pin 12):** when the application is locked, blink this LED every 300ms, otherwise power it off.
- **Warning LED (pin 11):** when the application is not locked, blink the LED with a rate proportional to the measured distance (rate between 0 and 1600ms for 0-400cm range). The closer the obstacle, the faster the blink. When the application is locked, blink the warning LED with the error LED (300 ms).
- **Light LED (pin 10 - PWM):** increase the brightness of the LED if the luminosity is decreasing. Same behavior for locked and unlocked mode.

- **LCD screen:** if the app is locked, print an error message which overrides any other message. If not locked:
    - Print the distance (cm or inches, default to cm) and a warning message if distance between 10-50cm.
    - On a different screen (toggled by the IR remote controller), print a message to reset default settings.
    - And on a third screen, print the luminosity obtained from the photoresistor.
- **IR remote controller:**
    - PLAY: if app is locked, also unlock the app (in addition to the push button)
    - UP/DOWN: toggle between the 3 screens of the LCD
    - EQ: change the default setting for distance unit (cm/inches) and save into EEPROM for the next boot
    - OFF: when the LCD is on the “reset settings” screen, press this button to reset all default settings (for now only the distance unit)
- **Photoresistor:** read the luminosity every 100ms. Set the brightness of the light LED accordingly, and print the luminosity on the screen if the LCD is on the third screen.

Come back to those steps at any moment if you have a doubt (+ rewatch the demo video).

Project Steps

I’ve divided the project into 11 steps, trying to make each step as small as possible, so you can see that any big project can be divided into smaller (and easier) units of work.

Here are the chronological steps I am going to take for the solution. The order for the steps has been chosen so each step builds on top of the previous one, and you can get a concrete result for each step.

- **Step 0-A (application design):** It’s best if you first try to imagine the steps first by yourself before seeing my solution (don’t write code, just write what steps you need to take on a text document or piece of paper). This is an additional practice on how to design a project and break it down into smaller units of work.
- **Step 0-B (not on the videos):** Try to redo the circuit by yourself, following the instructions from the hardware setup videos. Add the components one by one, and test them with a basic code to make sure they work correctly.
- **Step 1:** get the distance from the ultrasonic sensor, using interrupts, and print on serial to start.
- **Step 2:** Make the warning LED blink. Change the blink rate depending on the measured distance (0-400 cm → 0-1600 ms).
- **Step 3:** Lock the application when the distance is below a certain threshold (10cm), using a global boolean variable for the lock. When the app is locked, make the error + warning LEDs blink at the same time, every 300ms.

- **Step 4:** Debounce the button with polling (not interrupts here), and check for when we release the button while the app is locked. In that case, unlock the app and come back to normal behavior.
- **Step 5:** Setup LCD screen and print a message for 1 second in the setup(), before starting the loop().
- **Step 6:** Print measured distance and warning message on the LCD when the app is not locked. If locked, print an error message that only disappears when the app is unlocked.
- **Step 7:** Setup IR remote controller (with version 3 of the library here, and I'll show how to do with version 2 at the end), and map the buttons we'll need for the project.
- **Step 8:** Create the switch for all commands coming from the remote controller. When pressing on PLAY while the app is locked, also unlock the app.
- **Step 9:** Change the distance unit when pressing on the EQ button, and also save it into EEPROM. In the setup(), read the value from EEPROM. Update the distance message accordingly on the LCD screen, using the correct unit.
- **Step 10:** Add different modes (screens) for the LCD screen. The first one is the "distance" mode we already have. Second is a screen with a message saying "press on OFF to reset settings". Toggle between the 2 screens with the UP/DOWN buttons. When on the "reset" screen, press on OFF to reset default settings, and print a confirmation message.
- **Step 11:** Read value from photoresistor. Set the brightness for the light LED depending on the luminosity. Add a 3rd screen on the LCD screen to print the current luminosity.
- **Bonus step:** make the application work on the simulation (Tinkercad) by using the version 2 of the IR remote library.

Get started!

Enough tips, now, it's time for you to start the project!

It's important that you really take enough time to work on the project, that's the only way you will make great progress with Arduino. However, don't stay stuck for too long (days, weeks) on a given problem. It's totally OK if you need to watch the solution at some point. In this case, what you can do is come back to this step a few days after and try to redo it without any help.

Don't hesitate to also come back to any previous section/lesson to better understand a concept/functionality/component.

The solution will help you to get unstuck between 2 steps, but not only. After you've finished the project, the solution will show you how someone else is doing the project, so you have 2 versions, yours and mine. Also, I've tried to use as many best practices as possible in the code, so it will give you insights on how to improve your own code as well as better understand the best practices for your future projects.