Introduction to Robotics: Homework

Andrei Dumitriu andrei.dumitriu@fmi.unibuc.ro

Last Updated: November 10th 2022

Latest homework: Homework #4

1 Description

General guidelines for the document and all homework:

- Each homework is on a different page, despite the remaining page space.
- You must come with the homework already implemented, both hardware and software (food casserole is a great option for carrying your project). Installing or uploading the code at the lab leads to deduction in points.
- Code must already be uploaded to Git. We will only take into account the last upload before the deadline, not before the presentation.
- Coding style is of utmost importance in this lab.
- Color coding is of utmost importance in this lab.
- Some homework might have hard and soft deadlines, but **only if explicitly specified**.

The homework's purpose is to help you learn and practice the knowledge gained in this lab. We place strict emphasis on **correctly implementing** the requirements, not just getting it done. If you struggle with it, do not hesitate to contact your Teaching Assistant or Andrei directly in Microsoft teams.

2 Homework #0

Deadline (hard): Sunday, 23rd of October, 23:59:59.

The first homework is easy, and consists of two parts:

1. Install arduino IDE: `https://www.arduino.cc/en/software`
2. Create a **public** Git repository , according to these specifications:
   - You can use this as an example (will be updated as we homework progresses): `https://github.com/Irikos/IntroductionToRobotics`
   - Repository must be name "IntroductionToRobotics" (or something very similar, but with a good reason)
   - Must include similar Readme.md with title, initial description and subsection for each homework (with specified details)
   - You are free (and encouraged) to be creative with your repository description. **However**, the freedom of creativity must not be confused with and excuse to be lazy. The requirements specified here and in the repository description are considered basic and must be met
   - You can use other options, like Gitlab, Bitbucket etc as long as it meets the same basic requirements
   - Use the "Turn in" button in the teams assingment when done. Include the git link (make sure it's public!) in the text box. DO NOT uploade a file with the link.

These must be completed until Sunday, 23rd of October, 23:59:59.

3 Homework #1

Deadline (hard): Your respective lab in the week of October 24-30.

As was described in the laboratory, the homework consists of the following:

- **Components**: RBG LED (1 minimum), potentiometers (3 minimum), resistors and wires (per logic)
- **Technical Task:** Use a separat potentiometer in controlling each of the color of the RGB led (**R**ed, **G**reen and **B**lue). The control must be done with **digital electronics** (aka you must read the value of the potentiometer with Arduino, and write a mapped value to each of the pins connected to the led.
- **Publishing task:** You must add the code to the Git repo and continue updating the readme with at least the following details (but feel free to be more creative):
  1. Task Requirements
  2. Picture of the setup
  3. Link to video showcasing functionality (I recommend youtube, but anything I can access is fine)
  4. Remember to publish the video in the correct orientation. Don't do this: `https://youtu.be/Y8H0PlUtcto`
  5. **Hand in the homework on MS teams when done - aka when git is up to date**
- **Coding task:** Coding style is of utmost importance. You must have a perfectly clean code in order to receive the maximum grade. Keep in mind that magic numbers are not accepted, although you might see them in the lab (as a trade-off for speed). Remember to be consistent in your style, check the style guide and the provided style document and use correct spacing.

  Example used in lab which would not yield the maximum grade: "analogWrite(ledPin, potValue / 4) or "voltage = potValue * 5.0 / 1023.0".

  Mistakes observed there:

  1. "potValue / 4" is not precise, since the value does not always split into 4 perfectly. Use the **map() function**
  2. We calculate the value inside the analogWrite function
  3. **potValue** and **potPin** are not ideal naming versions. What happens if you change your potentiomer with a slider? (a slider has the same output range, so no change to the code would be needed - if written correctly)
  4. **4** is a magic number. **5.0** and **1023.0** are magic numbers. They should be replaced with constants, such as **maxAnalogValue**, **maxVoltageValue** etc

4 Homework #2

Deadline (hard): Your respective lab in week October 31 - November 6. You must have the Git ready and the assignment turned in before the lab.

As was described in the course, the homework consists of the following:

- **Components**: 5 LEDs, 1 button, 1 buzzer, resistors and wires (per logic)
- **General description:** Building the traffic lights for a crosswalk. You will use 2 LEDs to represent the traffic lights for people (red and green) and 3 LEDs to represent the traffic lights for cars (red, yellow and green). See the states it needs to go through. If anything is not clear, ask. Also, see the uploaded video (the intervals are different, but the states flow is the same). It's a traffic lights system for people and cars - don't overthink it.

  The system has the following states:

  1. **State 1** (default, reinstated after state 4 ends): green light for cars, red light for people, no sounds. Duration: indefinite, changed by pressing the button.
  2. **State 2** (initiated by counting down 8 seconds after a button press): the light should be yellow for cars, red for people and no sounds. Duration: 3 seconds.
  3. **State 3** (initiated after state 2 ends): red for cars, green for people and a beeping sound from the buzzer at a constant interval. Duration: 8 seconds.
  4. **State 4** (initiated after state 3 ends): red for cars, **blinking green** for people and a beeping sound from the buzzer, at a constant interval, faster than the beeping in state 3. This state should last 4 seconds.

  Be careful: pressing the button in any state other than state 1 should NOT yield any actions.

- **Publishing task:** You must add the code to the GitHub repo and continue updating the readme with at least the following details (but feel free to be more creative). I recommend using dropdowns in the readme, for each project, if you feel the readme is too cluttered:

  `https://gist.github.com/citrusui/07978f14b11adada364ff901e27c7f61`

  1. Task Requirements
  2. Picture of the setup
  3. Link to video showcasing functionality (I recommend youtube, but anything I can access is fine)

4. Remember to publish the video in the correct orientation. Don't do this: `https://youtu.be/Y8H0PlUtcto`
5. **Hand in the homework on MS teams when done - aka when Git is up to date**

- **Coding task:** Coding style is of utmost importance. You must have a perfectly clean code in order to receive the maximum grade. An important coding challenge present in this task is using **millis()** instead of **delay()**.

  However, remember that the bulk of the grade is if the projects **works**. Do not tend to small details unless you have the entire functionality done.

  Addendum: using interrupts yields a bonus, but it is not a must. Be careful, however: if you use interrupts, it must be properly coded (with debounce etc); coding it wrong can result in subtracted points (but we do appreciate any attempt, even unsuccessful ones).

5 Homework #3

Deadline (hard): Your respective lab in week November 7 - November 13. You must have the Git ready and the assignment turned in before the start of lab.

As was described in the course, the homework consists of the following:

- **Components**: 1 7-segment display, 1 joystick, resistors and wires (per logic)
- **General description:** You will use the joystick to control the position of the segment and "draw" on the display. The movement between segments should be natural (meaning they should jump from the current position only to neighbors, but without passing through "walls".

  The system has the following states:

  1. **State 1** (default, but also initiated after a button press in State 2): Current position blinking. Can use the joystick to move from one position to neighbors. Short pressing the button toggles state 2. Long pressing the button in state 1 resets the entire display by turning all the segments OFF and moving the current position to the decimal point.
  2. **State 2** (initiated after a button press in State 1): The current segment stops blinking, adopting the state of the segment before selection (ON or OFF). Toggling the X (or Y, you chose) axis should change the segment state from ON to OFF or from OFF to ON. Clicking the joystick should save the segment state and exit back to state 1.

  Be careful:

  1. Long pressing the button to reset should only be available in State 1.=
  2. Joystick movements should be done with toggle, as in the lab (joyMoved, etc)
  3. Correctly using interrupts with debounce yields a bonus, but it is not a must. **Code for homework with bonus points should be submitted before the course, not the lab.**
  4. The code in the lab for joystick movement is not perfect and can be improved. The state change is missing something (sounds familiar?), the code should be wrapped in a function and the constraint between 0 and 9 can be simplified.

Current segment	UP	DOWN	LEFT	RIGHT
a	N/A	g	f	b
b	a	g	f	N/A
c	g	d	e	dp
d	g	N/A	e	c
e	g	d	N/A	c
f	a	g	N/A	b
g	a	d	N/A	N/A
dp	N/A	N/A	c	N/A

- **Publishing task:** You must add the code to the Github repo and continue updating the readme with at least the following details (but feel free to be more creative). I recommend using dropdowns in the readme, for each project, if you feel the readme is too cluttered:

  `https://gist.github.com/citrusui/07978f14b11adada364ff901e27c7f61`

  1. Task Requirements
  2. Picture of the setup
  3. Link to video showcasing functionality (I recommend youtube, but anything I can access is fine)
  4. Remember to publish the video in the correct orientation. Don't do this: `https://youtu.be/Y8H0PlUtcto`
  5. **Hand in the homework on MS teams when done - aka when Git is up to date**

- **Coding task:** Coding style is of utmost importance. You must have a perfectly clean code in order to receive the maximum grade. An important coding challenge present in this task is using **millis()** instead of **delay()**.

  However, remember that the bulk of the grade is if the projects **works**. Do not tend to small details unless you have the entire functionality done.

6 Homework #4

Deadline (hard): Your respective lab in week November 14 - November 20. You must have the Git ready and the assignment turned in before the start of lab. The bonuses only apply if you upload the code before the course.

Basic requirements:

- **Components:** a joystick, a 4 digit 7-segment display, a 74hc595 shift register
- **General Description:** Use the joystick to move through the 4 digit 7 segment displays digits, press the button to lock in on the current digit and use the other axis to increment or decrement the number. Keep the button pressed to reset all the digit values and the current position to the first digit in the first state.

  The system has the following states:

  1. **First state:** you can use a joystick axis to cycle through the 4 digits; using the other axis does nothing. A blinking decimal point shows the current digit position. When pressing the button, you lock in on the selected digit and enter the second state.
  2. **Second state:** in this state, the decimal point stays always on, no longer blinking and you can no longer use the axis to cycle through the 4 digits. Instead, using the other axis, you can increment on decrement the number on the current digit IN HEX (aka from 0 to F, as in the lab). Pressing the button again returns you to the previous state. Also, keep in mind that when changing the number, you must increment it for each joystick movement - it should not work continuosly increment if you keep the joystick in one position (aka with joyMoved).
  3. Reset: toggled by long pressing the button **only in the first state**. When resetting, all the digits go back to 0 and the current position is set to the first (rightmost) digit, in the first state.

- **Publishing task:** You must add the code to the Github repo and continue updating the readme with at least the following details (but feel free to be more creative):

  1. Task Requirements
  2. Picture of the setup
  3. Link to video showcasing functionality (I recommend youtube, but anything I can access is fine)
  4. Remember to publish the video in the correct orientation. Don't do this: `https://youtu.be/Y8H0PlUtcto`

5. **Hand in the homework on MS teams when done - aka when github is up to date**

Useful links, Interrupts:

1. https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
2. https://create.arduino.cc/projecthub/rafitc/interrupts-basics-f475d5
3. https://www.youtube.com/watch?v=QtyOiTwOoQc
4. https://www.google.com/search?q=arduino+uno+interrupts&oq=arduino+uno+interrupts&aqs=chrome..69i57j0i67j0i512l7.5123j0j7&sourceid=chrome&ie=UTF-8

See the video on MS teams for an basic example. Differences between the homework requirements and the video:

1. You must use shift register
2. You must cycle all the way up to "F" in HEX
3. You need to add the "reset" functionality by keeping the button pressed

Have fun!