**Tea Machine**

Design with Microprocessors Project

Student: MARA OPRISOR

Technical University of Cluj-Napoca

2024-2025

Contents

[1. Introduction 3](#_Toc186214190)

[2. Bibliographic research 3](#_Toc186214191)

[2.1. Similar projects 3](#_Toc186214192)

[2.2. Components 4](#_Toc186214193)

[3. Proposed solution and implementation 5](#_Toc186214194)

[3.1. Components 5](#_Toc186214195)

[3.2. Software Part 6](#_Toc186214196)

[4. Testing and validation 8](#_Toc186214197)

[5. Conclusion 9](#_Toc186214198)

1. Introduction

This project implements an automated tea brewing system using an Arduino. It ensures that the tea bag is extracted from the water after the right time has passed. Also, the time that the tea bag should be let in the hot water is automatically computed based on the quantity of water.

Being a frequently tea drinker, I know how unpleasant it is to forget the tea bag in the water way longer than needed and they taste too bitter. Therefore, by designing this automated device, I will never have to worry about forgetting to remove the tea bag and getting a tea whose flavor is not the desired one.

The computation of the time based on how full is the teacup is a personal touch. We all know that if the cup is full and we keep the tea bag for 2-3 minutes, the flavor will be quite weak, while if the cup is half full that time would be perfect for the desired flavor.

1. Bibliographic research
   1. Similar projects

Some similar projects that I found on the internet are:

* <https://www.instructables.com/Automatic-Tea-Maker/>
* <https://www.instructables.com/Arduino-Automated-Tea-Dunker/>
* <https://www.digitaljunky.io/make-an-automated-tea-steeper-with-arduino/>

As expected, each one has a personal touch, but the principal is quite similar. In the next table I will present the similarities and differences between them in the next table.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Project 1 | Project 2 | Project 3 |
| Arduino Board | Arduino Mini | Arduino Nano | Arduino Uno |
| Main Action | Use a servo motor to lower and raise the tea bag | | |
| Start of Simulation | Uses a button to dunk the tea bag | Uses an ultrasonic sensor that detects the cup in order to dunk the tea bag | Uses a button to dunk the tea bag |
| Set the Time | Uses a potentiometer to set the time | Time cannot be changed | Time cannot be changed |
| Servo action | Lowers the bag, waits for the time to pass and only then raises it | Lowers and raises the teabag continuously | Lowers and raises the teabag continuously |
| Feedback Features | LCD that tracks the remaining time | Buzzer and LEDs for status feedback | Buzzer and LED indicators |

Table . Comparison between the 3 projects

* 1. Components
     1. LCD display

There are 2 types of LCD displays: the classical version or the I2C version.

|  |  |  |
| --- | --- | --- |
| Feature | Classical LCD | I2C LCD |
| Number of rows and columns | Both are 16x2 | |
| Number of pins | 7 or 8 pins | 4 pins |
| Control | Parallel communication | Serial communication (I2C protocol) |
| Library | LiquidCrystal library | LiquidCrystal\_I2C library |
| Ease of Use | Requires more setup | Easier, fewer connections |
| Cost | Generally cheaper | Slightly more expensive due to I2C module |

Table . Comparison between the two types of LCDs

Therefore, I2C LCDs simplify wiring and programming, making them ideal for projects with limited pins.

* + 1. Motors

There are 3 types of motors: DC, Servo and Stepper motors.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | DC Motor | Servo Motor | Stepper Motor |
| Type of motion | Continuous rotation | Precise angle control | Precise stepwise rotation |
| Control | Simple (speed via PWM) | Requires PWM for angle positioning | Requires stepper driver or pulses |
| Accuracy | Low | Defined angle range: 0°–180° | Very high |
| Use Case | Fans, wheels, etc. | Robotics, position control | CNC machines, 3D printers |
| Cost | Low | Moderate | Moderate to High |

Table . Comparison between types of motors

The servo motor is ideal for a project like this because it offers precise angle control. Its simplicity in control and moderate cost make it ideal for small, automated tasks like this.

* + 1. Load Cell

There are multiple types of load cells. I will present the similarities and differences between the 1 kg, 5 kg and 10 kg load cells.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | 1kg Load Cell | 5kg Load Cell | 10kg Load Cell |
| Capacity | Up to 1kg | Up to 5kg | Up to 10kg |
| Output signal | Millivolts | | |
| Sensitivity | Very sensitive | |  | | --- | | Moderate sensitivity |  |  | | --- | |  | | |  | | --- | | Moderate sensitivity |  |  | | --- | |  | |
| Size | Compact | Medium | Larger |

Table . Comparison between types of load cells

The 1kg load cell is best for my project because it is designed for measuring small weights with high precision, making it ideal for my system where the liquid's weight changes are subtle.

* + 1. HX711 Amplifier
* Purpose: Amplifies the small voltage signal from the load cell for the Arduino to read.
* Resolution: 24-bit ADC for high precision.
* Interface: Digital.
* Power Supply: Operates on 2.7V–5V, low power consumption.

1. Proposed solution and implementation
   1. Components

For my tea machine project, I will use the following components:

* Arduino Uno board
* Servo Motor MG90S
* 1kg Load Cell and HX711 Amplifier
* LCD screen
* Push buttons

A circuit board with wires

Description automatically generated

Figure . Hardware scheme of the project

* 1. Software Part

In order to make debugging easier, the code is divided into clear parts and functions as follows:

* Library declaration part:
  + The libraries that are needed for the components to work are declared here.
  + These are:
    - Servo.h
    - LiquidCrystal\_I2C.h
    - Wire.h
    - HX711.h
* Constants declaration part:
  + It makes the code easier to be read and mistakes are avoided.
  + Here are declared the pins that are used for each component, making the task of changing them (if we decide to) really easy.
  + Here are also declared some constants (weight of an empty cup, weight of a full cup, the wait time for the tea). Therefore, if we change the cup or the tea, we can change it really easy.
* Object creation and variables declaration:
  + Here we create the objects for the servo, lcd and scale.
  + Also, here we declare the variable that are used in code and are initialized if needed.
* setup() function:
  + It helps us prepare the simulation.
  + We attach the corresponding pin to the servo.
  + We initialize the LCD and we print the “Press Start” message on it.
  + We initialize the scale and we calibrate it.
* loop() function:
  + It represents the actual simulation.
  + The simulation works as a finite state machine that has 5 states: idle, compute, wait for confirm, brew and complete.
  + Each state is created as a separate function that handles what happens during that state.
* handleIdleState() function:
  + If the start button is pressed, the simulation moves on to the next state.
* handleComputeState() function:
  + The message on the LCD is changed.
  + The weight of the cup is read and the wait time is computed based on this value.
* handleWaitForConfirmState() function:
  + If the start button is pressed the simulation moves on to the next state.
  + Before pressing the start button we can adjust the wait time using the increase and decrease buttons, the value being changed with ten seconds at each press (however, it cannot go under 10 seconds or over the standard wait time multiplied by 2).
* handleBrewState() function:
  + This function handles that the servo rotates to 180 such that the tea beg is placed in the water.
  + It waits until the waiting time has passed (the time that passes is shown on the LCD, such that the user is able to know how much does he have to wait).
  + When the time has passed, the servo returns to 0, such that the tea bag is removed from the water.
* handleCompleteState() function:
  + This function print on the LCD the message “Enjoy your tea!”.
  + After waiting for 5 seconds, the simulation returns to the idle state and we can start again the simulation.
* computeWaitTime(weight) function:
  + This function decide how much should the tea be kept in water.
  + If the cup is only half full or less, the tea should be kept half of the standard time.
  + Otherwise, the wait time is equal to the standard one.
* debounceButtons() function:
  + Handles the debouncing of the three buttons.

1. Testing and validation

When I first started to implement the project, I started first testing each component individually.

1. The LCD: It was easy to test, I just had verified that the text was printing correctly on it.
2. The servo motor: I had to see how exactly was it working and how its angles were corresponding to its movement. I had to verify which value should be written when the tea bag had to be put in water and the value that should be written when it had to be extracted from water.
3. The load cell and hx711 amplifier: This was the most complex component of my project.
   1. Before starting to use it, I had to create a platform for it.
   2. Then, I had to find the calibration factor. For this, I had to write a separate program that starts from a random calibration factor. I had to get an object whose weight I knew and to change the calibration factor until the value measured by the load was the same as the known value of the weight.
   3. This was followed by finding the weight of the empty cup but also of the full cup, values that were determined by this sensor and used in the code.

After testing each component, I started putting everything together and writing the code for the device as a whole. The first problem I encountered was the fact that the buttons were not working as expected. They were reacting only once in many pushes or increasing the value to many times in one push. This was easy to fix by adding debouncers on all the buttons.

The next problem I encountered was that the simulation was sometimes skipping steps or entering deadlocks. This problem generated the idea of implementing my project as a finite state machine that goes smoothly from one state to another.

The last problem was that of arranging the components such that the teabag enters the cup, how long should the string of the tea bag be or how to tie the tea bag on the arm of the device. This needed many tries and only when I finished those tests, was I able to place the pillar into a permanent position.

1. Conclusion

Even though the project started as some separate components, all working properly but independent of each other, I was able to design a tea machine that waits for inputs from the user that can use one of the 3 available buttons, executes some operations in the background (finds how much water is it in the cup, decides how much should the tea bag should be let to sit in water, controls the servo motor etc.) and gives feedback to the user on the LCD screen.

The code is really easy to adapt. The constants are all declared in the beginning using “#define”, therefore if changing the cup or if you do not like the standard brewing time for the tea, these values can be changed fast by replacing the old values with new ones.

Some improvements that could be made would be:

* Adding a temperature sensor for the water in order to assure that the water has the perfect temperature.
* The prototype for the device could be changed in order to give it a more beautiful appearance.
* The Arduino Uno could be replaced with an Arduino Nano to make the design smaller and easier to integrate into an actual device.
* The wires could be soldered to ensure a secure and stable connection, reducing the risk of them becoming loose during operation.

A coffee cup with wires and a digital meter

Description automatically generated with medium confidence

Figure . Final look of the project