TEMPERATURE SENSOR

EMBEDDED SYSTEM 3°A

TEAM:

RUBI ABIGAIL NOH MANZANERO	1609092
ALDO MARTIN MEX SIERRA	1609080
GISELLE GUADALUPE VALDÉS RODRIGUEZ	1709154
SUSANA ARJONA SANSORES	1709005

TEACHER: LIC. VÍCTOR JOSÉ RAMÍREZ CASTAÑÓN

Coding Standard.

Terminology

For those unfamiliar with CSS terminology, these are the concise terms used in these standards.

selector { property: value;}

A rule set (also called a rule) consists of a selector followed by a declaration block.

The selector consists of everything up to (but not including) the first left curly brace ({).

A declaration block starts with a left curly brace ({) and ends with the matching right curly brace (}). In between there must be a list of zero or more semicolon-separated (;) declarations.

A declaration consists of a property name, followed by a colon (:), followed by a value.

Whitespace

Indentation

Do not use tabs for indentation. They lead to inconsistent display of the source code, since many text editors and most text viewers (like web browsers) cannot have their "tab size" configured.

Use 2 spaces for each level of indentation, the same standard as Drupal's PHP and JavaScript code.

- Declarations (property/value pairs) should be indented one level relative to their selector.
- Rulesets within a media block or a media query should be indented one level relative to the media statement.
- Comments should be indented the same amount as the declaration or ruleset they describe.

```
@media print {
        /* This line is indented with 2 spaces,
2 spaces x 1 level of indentation. */
        .example {
                /* This line is indented with 4
spaces, 2 spaces x 2 levels of
indentation. */
                padding: 0;
        }
}
```

Blank lines

- In general, do NOT separate each ruleset by a blank line.
- If a ruleset has a proceeding Doxygen-style or single-line-style comment that describes it, place a blank line before the comment.
- If two rulesets have no interleaving blank line, they must be logically related. If they are not logically related to each other, add a blank line and a comment describing the second ruleset.

```
/* A comment describing the ruleset. */
.selector-1,
.selector-2,
.selector-3[type="text"] {
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;
box-sizing: border-box;
display: block;
margin: 0;
font-family: Times, "Times New Roman",
sans-serif;  color: #333;
background: #fff;
background: linear-gradient(#fff,
rgba(0, 0, 0, 0.8));
}
/**
* A longer comment describing this
ruleset. Note
* the blank line before the docblock.
*/
.selector-4,
.selector-5 {
background-color: lime;
}
/* This logical grouping of rulesets has no interleaving blank lines. */
```

```
.profile {
margin: 16px 0;
margin: 1rem 0;
}
.profile__picture {
  float: right; /* LTR */
}
```

Line endings

There MUST NOT be any whitespace (spaces or tabs) at the end of lines. This means blank lines should also not contain any spaces or tabs. Inconsistent trailing whitespace can add lines to diffs/patches and makes changes harder to notice.

All text files should end with a single blank line. This makes git commits easier to read since it's clearer what is being changed when lines are added to the end of a file and it avoids the verbose \ No newline at end of file warning in patches.

Files should be formatted with Unix line endings (a newline character, denoted as \n or LF), which is also the default in Mac OS X. Do not use Windows line endings (a carriage return plus a newline, denoted as \r\n or CRLF).

Tip: configure your editor to “show invisibles”. This will allow you to eliminate end-of-line whitespace, eliminate unintended blank-line whitespace, and avoid polluting commits.

Drupal 8 includes an EditorConfig file in its root directory to help maintain these whitespace conventions.

Comments

Well commented code is extremely important. Take time to describe components, how they work, their limitations, and the way they are constructed. Don't leave others guessing as to the purpose of uncommon or non-obvious code.

To stay consistent with the rest of Drupal's code base, we borrow some of the CSS comment styles from the Doxygen and comment formatting conventions for PHP files.

File comments

Each file should start with a comment describing what the file does. For example:

```
/**
* @file
* Short description describing the file.
*
```

** The first sentence of the long*

description starts here and continues on

this

** line for a while finally concluding*

here at the end of this paragraph.

**/*

Note that a blank line should follow a file comment. And keep line-lengths to 80 columns, when possible. For more information, see the PHP file comment standards.

Multi-line comments

When describing a ruleset or set of rulesets, any comment that requires 2 or more lines (wrapped to 80 characters) must follow the Doxygen comment style (also called a "docblock").

*/***

** Short description using Doxygen-style*

comment format.

***

** The first sentence of the long*

description starts here and continues on

this

** line for a while finally concluding*

here at the end of this paragraph.

***

** The long description is ideal for more*

detailed explanations and

** documentation. It can include example*

HTML, URLs, or any other information

** that is deemed necessary or useful.*

**/.example-rule {*

}

Place the comment on the line immediately above the ruleset (or rulesets) it describes. Place a blank line before the docblock comment. See the Doxygen and comment formatting conventions for more info.

Single-line comments

When describing a property or ruleset, any comment that can be written inside the 80 character line length limit can use a simple CSS comment style.

```
.example {
/* Override the default margins. */
margin: 0;
}
/* This is a variant of the .example
component. */
.example--item {
display: inline;
}
```

Place the comment on the line immediately above the property or ruleset it describes. The comment should be indented the same amount as the property or ruleset it describes.

If the comment is describing a ruleset, place a blank line before the comment.

Styling for Right-To-Left Languages

It is common for RTL language websites to have their designs flipped in the left/right direction. For direction specific property/values, add the comment /* LTR */ on the same line preceded by a single space. In Drupal 6 and 7, the inclusion of a separate RTL stylesheet is automated. In Drupal 8, follow with an additional ruleset containing the inverse property/values, beginning with the attribute selector [dir="rtl"].

Example Rulesets for Drupal 6 and Drupal 7

[example.css]

```
.example-rule {
float: left; /* LTR */
margin-right: 24px;
margin-right: 1.5rem; /* LTR */
padding: 0 4px;
padding: 0 0.25rem;
}
```

[example-rtl.css]

.example-rule {

float: right;

margin-left: 24px;

margin-left: 1.5rem;

margin-right: 0;

}

Example Rulesets for Drupal 8

[example.css]

.example-rule {

float: left; / LTR */*

margin-right: 24px;

margin-right: 1.5rem; / LTR */*

padding: 0 4px;

padding: 0 0.25rem;

}

[dir="rtl"] .example-rule {

float: right;

margin-right: 24px;

margin-left: 1.5rem;

margin-right: 0;

}

- when you use the keywords, 'left' or 'right' in a property, e.g. float: left;
- where you use unequal margin, padding or borders on the sides of a box, e.g. margin-left: 1rem; or padding: 0 0 0 2rem;
- where you specify the direction of the language, e.g. direction: ltr;

Format

Our CSS formatting ensures the code is easy to read, easy to clearly comment, minimizes the chance of accidentally introducing errors, and results in useful Git diffs and blames.

Rulesets

- Use one selector per line when a ruleset has a group of selectors separated by commas.

- The opening brace ({) of a ruleset's declaration block should be on the same line as the selector (or the same line as the last selector in a group of selectors.) The opening brace should include a single space before it.
- Place the closing brace (}) of a ruleset in the same column as the first character in the selector of the ruleset.
- Include one declaration per line in a declaration block.
- Each declaration should be indented one level relative to its selector.

Example Ruleset

.selector-alpha,.

selector-beta {

counter-reset: section;

text-transform: small-caps;

}

Properties

- In a declaration, the property name should be immediately followed by a colon, then a single space, and then the property's value.
- Include a semi-colon at the end of all declarations, including the last declaration in a declaration block.
- When hex values are used for colors, use lowercase and, if possible, the shorthand syntax, e.g. #aaa. Colors may be expressed with any valid CSS value, such as hex value, color keyword, rgb() or rgba(). Note that IE8 does not support all color syntaxes and will require a fallback value.
- For property values that require quotes, use double quotes instead of single quotes, e.g. font-family: "Arial Black", Arial, sans-serif; and content: " ";.
- If a property does not require quotes (e.g. url(), do not add them. This means background-image: url(path/image.png) instead of background-image: url("path/image.png")
- Use rem units preceded by px units for a safe fallback, unless it creates an undesired effect.
- Quote attribute values in selectors, e.g. input[type="checkbox"].
- Where allowed, avoid specifying units for zero-values, e.g. use margin: 0;instead of margin: 0px;.
- Include a space after each comma in comma-separated property or function values.
- Do not use spaces around the parentheses in a function, e.g. color: rgba(0, 0, 0, 0.8);
- Use lower case function names, correct: color: rgba(0, 0, 0, 0.8); incorrect: color: RGBA(0, 0, 0, 0.8);

Example Properties

display: block;	Basic syntax
color: #fff color: #df7dcf	Use shorthand syntax for hexadecimal colors when possible Always use lowercase
font-family: "Frutiger Ultra"	Use double quotes instead of single quotes
text-shadow: 0 0 2px #ddd	Do not attach units to zero-values

font-size: 24px; font-size: 1.5rem;	Use rem units preceded by px units for a safe fallback, unless it creates an undesired effect.
color: rgba(0, 136, 18, 0.8)	Spaces MUST follow commas in property or function values

Declaration order

The declarations in a ruleset should be ordered so that the purpose of the declaration block is most obvious. Clarity should be the guiding principle. We can help to achieve this goal by placing structurally important properties before others: positioning, box model, then other properties.

1. Positioning properties include: position, float, clear, top, right, bottom, left, direction, and z-index.
2. Box model properties include: display, [(max|min)-]height, [(max|min)-]width, margin, padding, border and their various longhand forms (margin-top, etc.) Plus box-sizing.
3. Other declarations.

Within each of the above groups, properties can be grouped alphabetically or grouped with like properties next to each other, e.g. putting font and text properties next to each other. Drupal's coding standards are purposefully vague here because there is no consensus on this issue (as of 2013), but we respect each other's abilities and preferences.

Vendor prefixed properties should be directly before their non-prefixed version. This allows the official version of the property to override any inconsistencies in the vendor-prefixed versions once those browsers implement the official property. If browser bugs or cross-browser issues necessitate any deviation from this ordering, it should be clearly documented.

Again, the order of properties is meant to reinforce the purpose of the ruleset. As such, it is much more important to add comments to the ruleset than to worry about property ordering.

```
.selector {
/* Positioning declarations */
position: absolute;
top: 0;
left: 0;  z-index: 10;
 /* Box model declarations */
display: inline-block;
width: 100%;
padding: 10px;
padding: 0.625rem;
border: 1px solid #333;
/* Other declarations */
background: #000;
```

color: #fff;

font-family: sans-serif;

font-size: 18px;

font-size: 1.125rem;

}

Tools like CSScomb may help with automating the order of properties (CSScomb settings for Drupal).

Exceptions and slight deviations

Large blocks of single declarations can use a slightly different, single-line format. In this case, a space should be included after the opening brace and before the closing brace.

.selector-1 { width: 10%; }

.selector-2 { width: 20%; }

.selector-3 { width: 30%; }

Long, comma-separated property values—such as collections of gradients or shadows—can be arranged across multiple lines in an effort to improve readability and produce more useful diffs.

.selector {

background-image:

linear-gradient(#fff, #ccc),

linear-gradient(#f3c, #4ec);

box-shadow:

1px 1px 1px #000,

2px 2px 1px 1px #ccc inset;

}

Media Queries

Media queries should be written in the same style as ruleset. Any containing rulesets are indented by two spaces.

- One space between the media feature and the value.
- All values to be written in rems unless it is inappropriate.
- Add the pixel value in a comment directly after the the opening brace.

@media screen and (min-width: 28.125rem)

{ / 450px */*

#page {

margin-left: 20px;

margin-left: 1.25rem;

margin-right: 20px;

margin-right: 1.25rem;

}

}

Miscellaneous

@charset statements

Character set statements (like @charset "UTF-8";) are only valid if they are at the very top of a CSS file. Since Drupal's CSS aggregator combines multiple CSS files into one file, Drupal will strip all @charset statements so that the aggregated file remains valid CSS.

This means CSS files MUST NOT include any @charset statements. The default encoding for CSS files is UTF-8. Any CSS comment or content property values MUST be encoded with UTF-8.

Methodology

SCRUM

We will use the agile method because it improves the situation with a developer its more clear and it gives more value in the process development, let's say it's an agile and easy way. Also, when it has short interactions with constant feedback it creates a bigger motivation, it makes the projects less monotone and more agile for the product we want to work with.

We used this methodology because we are a team of four members that work in a collective way and our project will last four about a month.

This methodology fits perfectly because the steps must be quick in intervals of time different, exact and organize. Our product owner is ourselves and we decide it to end with this project in less time because we have a small budget and we need to sell this program, this methodology benefit us, that's why we want to work with it.

Why we use it SCRUM?

We are a team of four people working in a collective way and our project will take one month approximately.

This methodology fits perfectly because the steps need to be so fast in different, exact and ordered intervals of time because our Product Owner is ourselves and we decided finish this work in less time because we have a short budget and we need to buy this program, also because the scrum master in

Other causes are that we want and need to be a highly productive team

We would use it in our project because scrum its center in adjusting the results and answer the real and exact requirements of the clients in this case our teacher, we will work with this:

- Definition of the project
- Users' stories
- Table
- Planning Poker
- Team Velocity
- Programing

These tools will function to us for having a good work because they are organizing and precisely.

Profiles

In a Scrum team several profiles have a roll:

Profiles	Definition	Who will working in this?
Product Owner	Is the person responsible of the success of the product from the point of view of the clients. This person defines the project objective and focus and maximize the cost effectiveness of this product.	The Lic. Victor Ramirez defines the product and the objective that we need to have, the cost effectiveness and the utility of the product. We choose the professor because he teaches us the learning objectives with acceptance criteria
The Scrum Master	It's the coach of the rest of the team and the one that helps to achieve the maximum level of productivity and guarantee the objectives.	Lic.Aldo Mex is the one in charge who knows if the product owner is going in a correct way.
The Scrum Team	Its form from all the necessary individual for the construction of the product realize a fundamental work: Chasing a common objective, with auto management and auto organize structure.	Lic. Rub Noh Lic. Aldo Mex Lic. Giselle Valdes Lic. Susan Arjona This people are working on the development of this project they define the results and the quantity of time its require.
The Stakeholders	It's the people that will receive some benefit from this project (merchants, owners, directors).	Computers Enterprise (HP, Lenovo, MSI, Apple, etc.)

The team members have different sets of abilities, they help each other, this way nobody becomes a leader in the delivery of the work. All the members help each other to secure a successful ending of the sprint.

How we use it in our work

1. We choose the team that has the skills and needs that our product required.

2. We choose a responsible person who has a clear vision of what they need, the possibility and if it is feasible.

3. Another aspect we chose was a Scrum Master that led us to the entire team through the SCRUM work system, helping the team eliminate everything that does not work for us, in this case Aldo Mex who had the responsibility of being our Scrum Master.

4. We use and elaborate a stimulation a list of objectives and slopes as shown in the example.

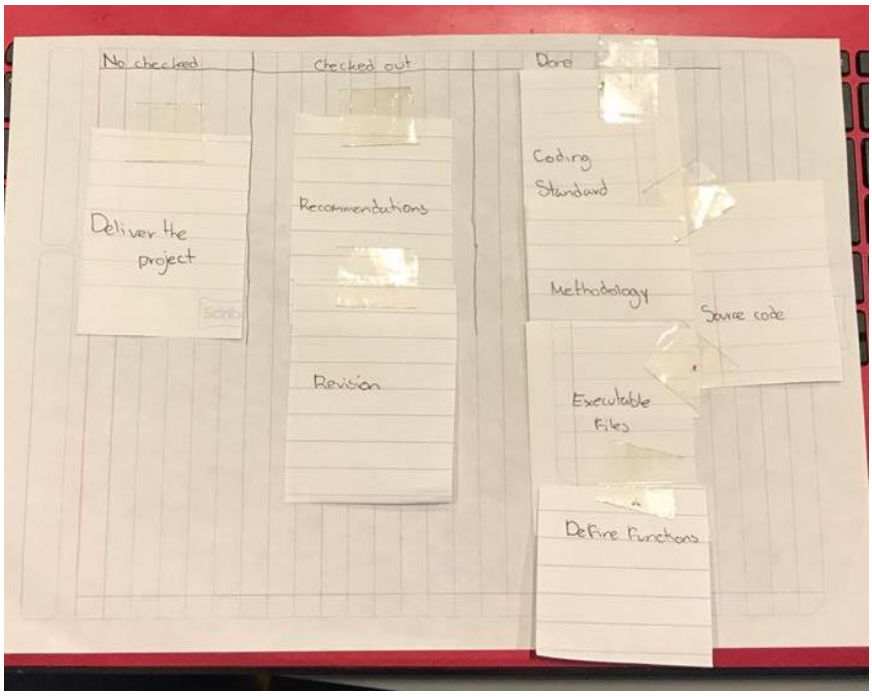
DATE	DEVELOPMENT ACTIVITY.	NOTES.	OBJECTIVES.
October 5, 2018	1.Coding standard.	1.Specify and argue.	Know the best standards you can use in the project.
October 8, 2018	2.Methodology.	1.Modify introduction. 2.be 2 sheets.	Know how to use Scrum in a project the advantages and disadvantages.
October 10, 2018	3.Source code.	1.Investigate more about the Pic.	Read and know about our program that we use.
October 12, 2018	4.Source code.	Be specific	Make it emit a sound in the code for warning.
October 15, 2018	5.Source code.	Show errors	
October 17, 2018	Executable files.		Describe how the code is used from scratch.
October 19, 2018	Recommendations.	Use the errors we had.	Write suggestions if you continue carrying out the project.
October 22, 2018	Revision.		
October 24, 2018	Project delivery.		

Accomplished

Corrections

5. Sprint planning, since we did a general review of what we had and always trying to increase and eliminate and all that was worked directly on GitHub.

6. In the same way we made a blackboard with three columns with the slope, process and done as shown in the image.



7. We met daily to see what was missing and always answering the same questions.

• What did we do yesterday to help the team finish what is missing?

• What are we going to do tomorrow to help the team?

• What obstacles stand in your way or that of the team?

These questions were answered in approximately 15 minutes or less

8. The review and demonstration, in which the team in general represent what has been done and what can be improved.

Recommendations.

The project “Temperature Sensor” it is designed with the objective of give security about the temperature which has a laptop in use and in this way don’t have an excess in this sense.

It was used a language similar to C++. The CSS COMPILER it is a compiler which translate the program write it in a language with a high level to assembly language.

Programming.

If the person wants to continue with this program, we suggest read the code with the respective comments because in this way you see what we did and how works each part of the project.

In our opinion this code is found correctly and specifically, but if you know more about the CSS COMPILER and you find another way easier of resume and abbreviate this code, you are invited to do it and similarly if you want to add innovations such as new functions or processes.

Problems programming.

The CSS COMPILER it´s a new program for us, we had to research about how was the kind of language and how it worked for that reason we had some problems at the time of haven´t an appropriate sequence and confuse “Devices and Fuses”.

Other problem was adding “Buzzer” because emitted a lineal noise nor sounding like a normal alarm for that reason the next thing that we add was “Delay” as a result of when this device emits the sound stop each second.

Technical issue.

It´s recommended be open to improvement and changes with the pass of the time with the microcontrollers and sensors, nowadays we use those who were in the market available like the sensor (LM35) and the microcontroller (18F4550).

The methodology employee was SCRUM because we can do the project with short phases of two until four weeks. This help us to adapt to the changes happened in the process namely allow a lot of flexibility at the time of committing an error and you can improve or change something in the middle of the project, since after each phase the tasks and objectives are rethought. Other methodologies did not cover our needs so if you want to continue carrying out this project, we suggest using SCRUM to facilitate the development of this project.

Diagram UML

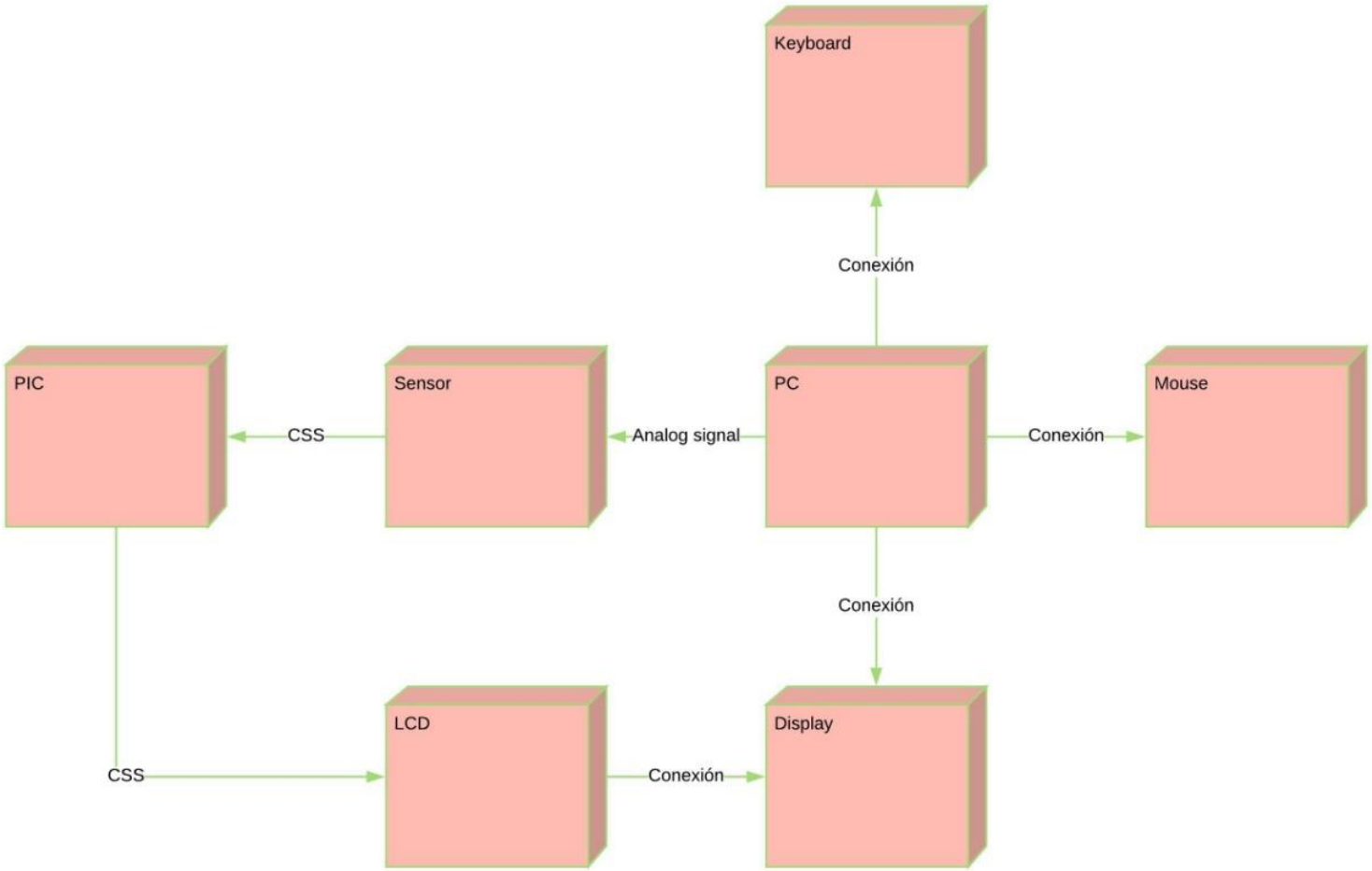


Figure 1: Deployment Diagram

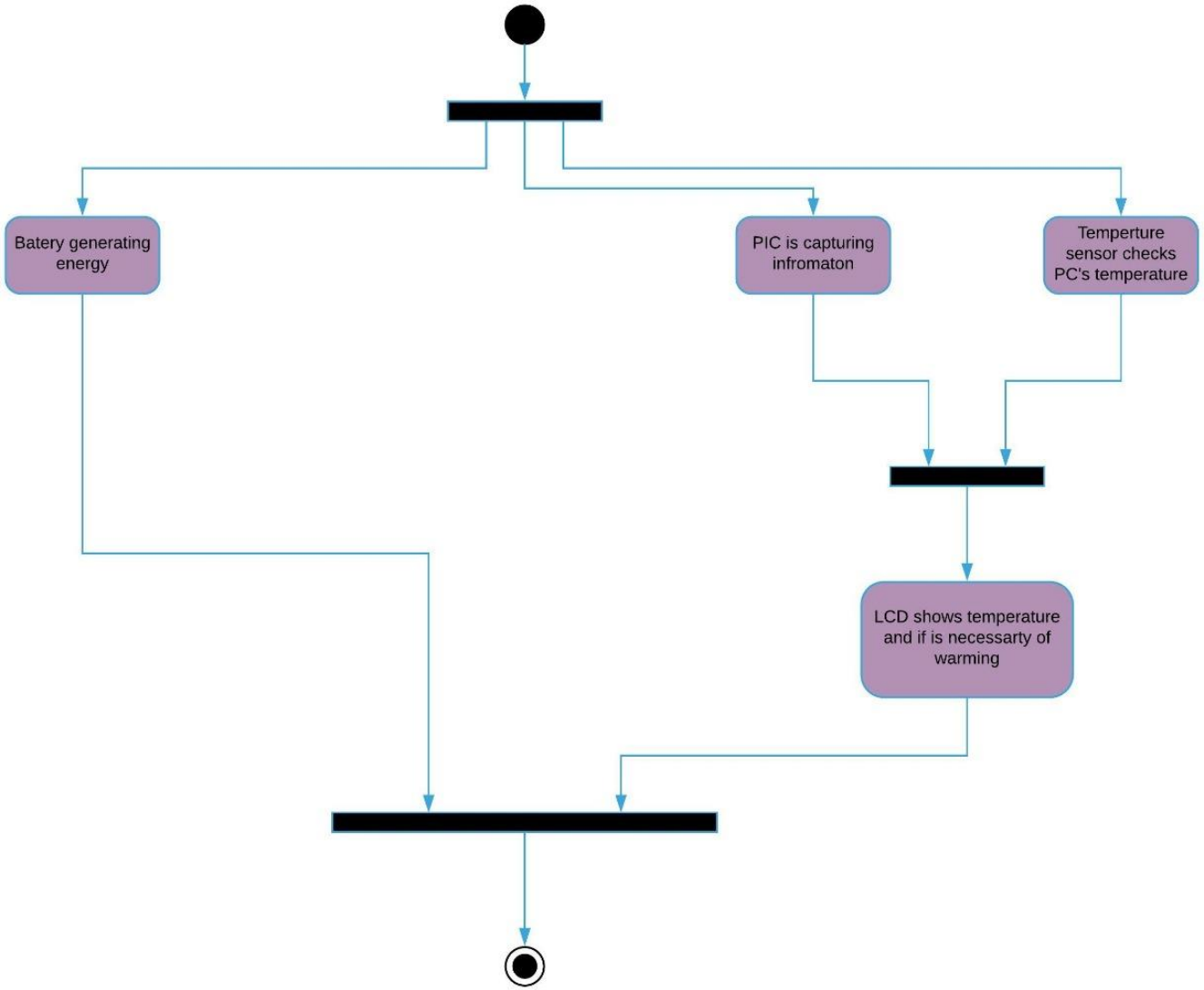
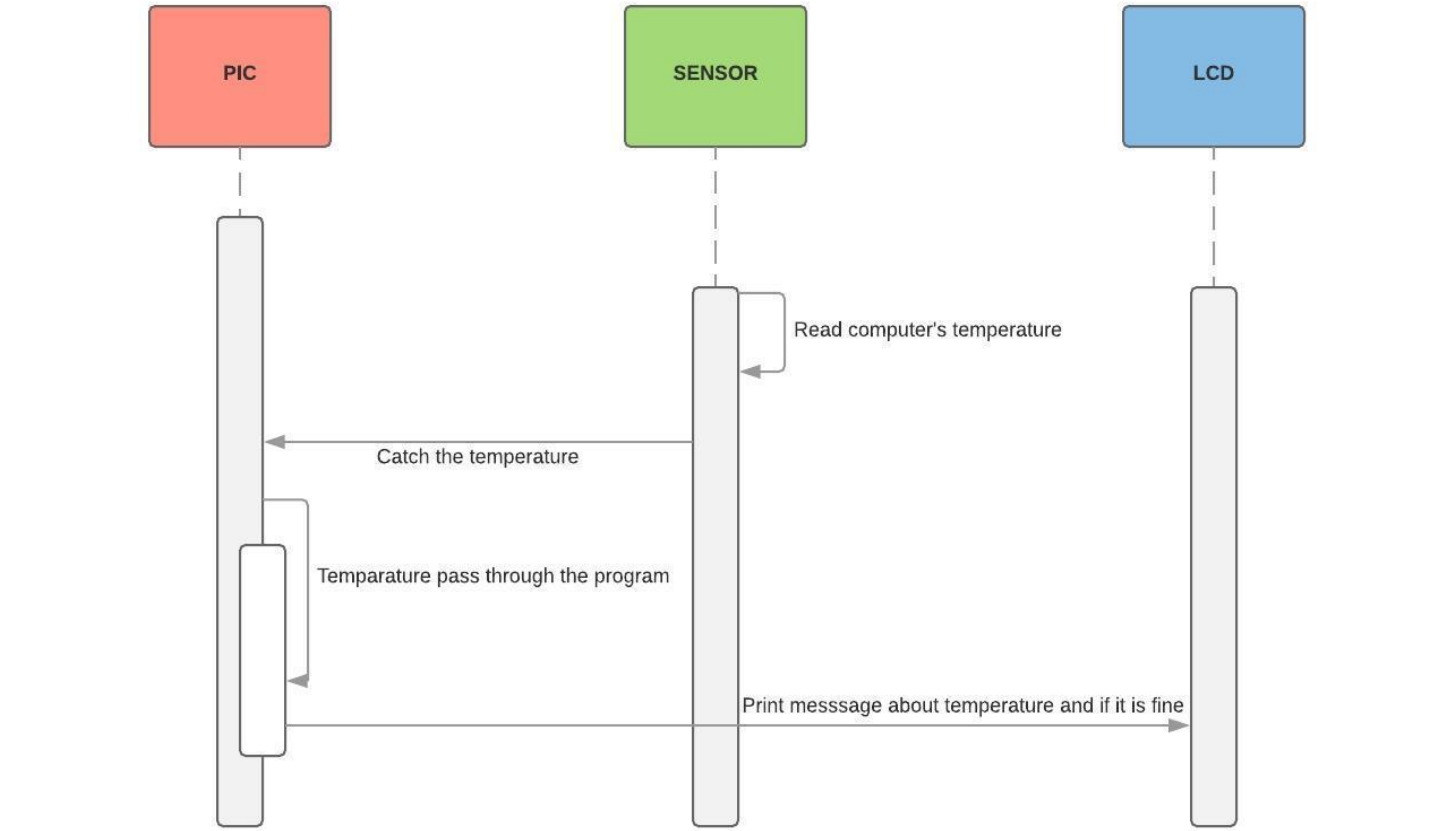


Figure 2: Activity Diagram



REFERENCE

https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/

https://www.drupal.org/docs/develop/standards/css/css-formatting-guidelines

http://www.aspl.es/fact/files/aspl-fact/estandares-node2.html

https://www.youtube.com/watch?v=7jJTASEtx_o