Lab #1

Group number: 17

Group members: Holmes Joseph, Akhil Maddikonda

1. Prelab

1.1. What flag for ls displays all files, even hidden ones?

⇒ls -a displays all the hidden files

1.2. How do you move to the parent directory of the current one?

⇒cd .. can be used to move to the current one, and just cd can move to the home directory

1.3. What is the difference between the mv and cp commands?

⇒mv is the move command used to move by removing files from the home directory, cp command copies the contents without removing them from the initial location

1.4. What does the -h flag of ls do? Hint: use man to find out.

⇒- h is used to make it to human understandable data sizes

1.5. In a git repository, what command displays whether there are any local changes and what they are?

⇒ git diff, gives one the changes that are made

1.6. In a git repository, what does git pull do?

⇒it makes any changes made in the desired repository in GitHub to an individual desired repository

2. Checkoffs

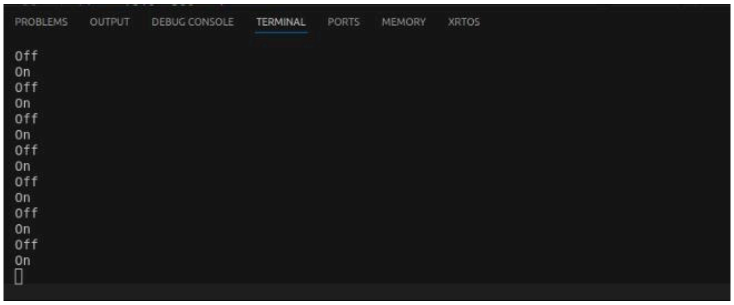
2.1. Verify that the VS Code and command-line mechanisms are working. Explain how the timing of the blinking of the LED is controlled.

⇒ The blinking of the LED is controlled by a **timer** that is set to periodically trigger an event, which has two inputs: an initial delay and a frequency of the counter.

2.2. Show your modified LF program. Explain how this use of timers is different from the sleep function used in the C code blink.c.

⇒ LF-timer allows for concurrency while the sleep function in the C program halts program execution and prevents other tasks from being handled concurrently.

2.3. Show ON-OFF output.

⇒ 

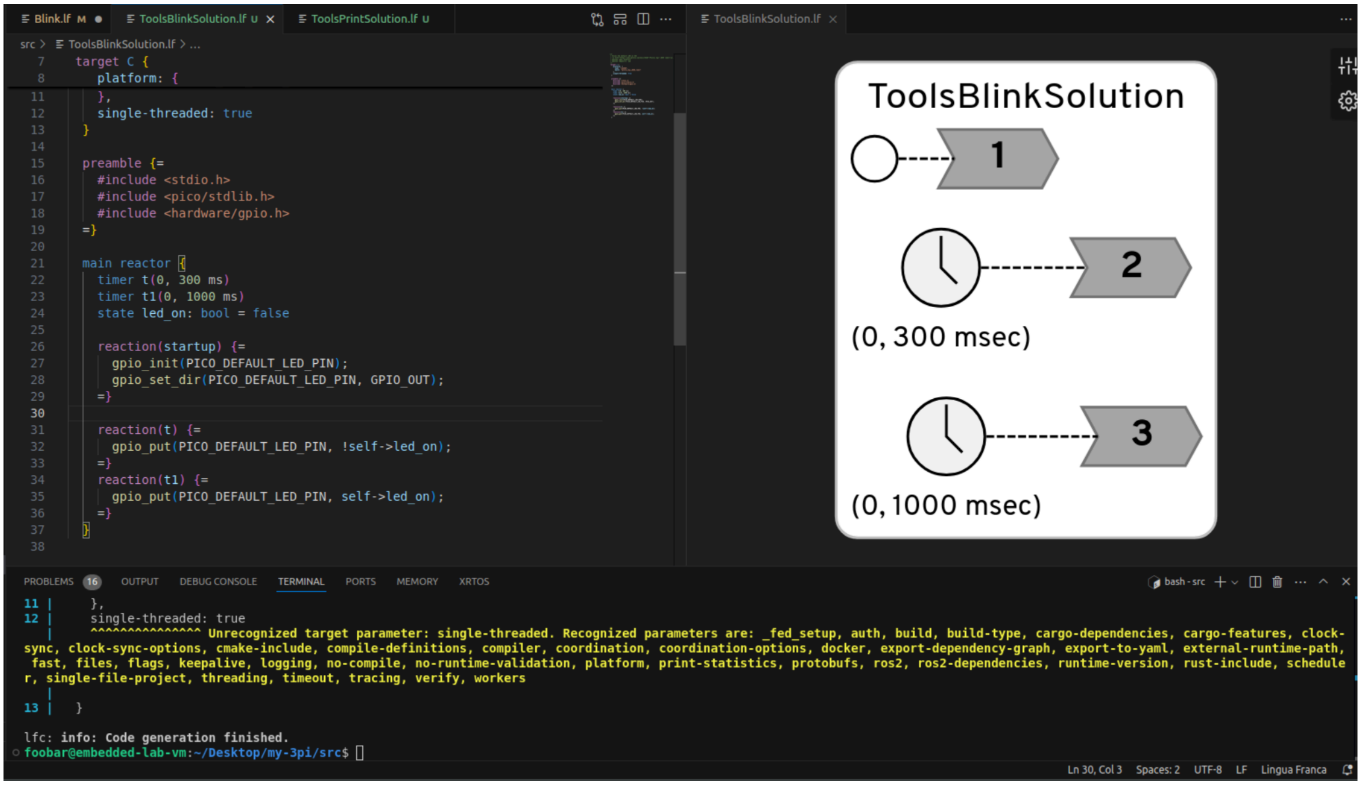
```
Off
On
Off
On
Off
On
Off
On
Off
On
Off
On
Off
On
```

2.4. Show and explain how your program works.

⇒

ToolsBlinkSolution.lf

This code has two timers; when the first timer triggers, it sets the LED_PIN to true, which is to switch the LED on, and when the second timer triggers, it sets LED_PIN to false, which is to switch the LED off.

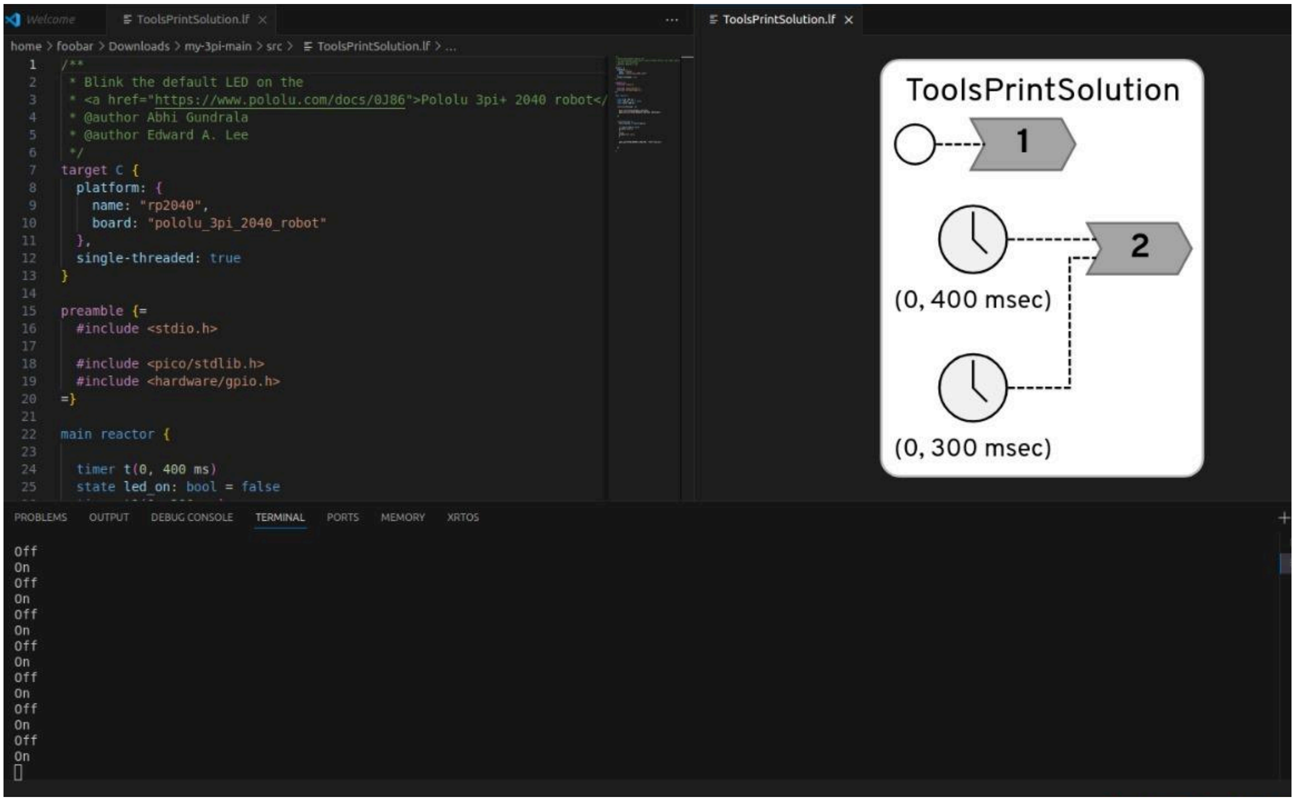


```
target C {
  platform: {
  },
  single-threaded: true
}

preamble {=
  #include <stdio.h>
  #include <pico/stdlib.h>
  #include <hardware/gpio.h>
=}

main reactor {
  timer t(0, 300 ms)
  timer t1(0, 1000 ms)
  state led_on: bool = false

  reaction(startup) {=
    gpio_init(PICO_DEFAULT_LED_PIN);
    gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);
  =}

  reaction(t) {=
    gpio_put(PICO_DEFAULT_LED_PIN, !self->led_on);
  =}
  reaction(t1) {=
    gpio_put(PICO_DEFAULT_LED_PIN, self->led_on);
  =}
}
```

```
11 |     },
12 |     single-threaded: true
   |     ^^^^^^^^^^^^^^^ Unrecognized target parameter: single-threaded. Recognized parameters are: _fed_setup, auth, build, build-type, cargo-dependencies, cargo-features, clock-sync, clock-sync-options, cmake-include, compile-definitions, compiler, coordination, coordination-options, docker, export-dependency-graph, export-to-yaml, external-runtime-path, fast, files, flags, keepalive, logging, no-compile, no-runtime-validation, platform, print-statistics, protobufs, ros2, ros2-dependencies, runtime-version, rust-include, scheduler, single-file-project, threading, timeout, tracing, verify, workers
   |
13 |   }

lfc: info: Code generation finished.
foobar@embedded-lab-vm:~/Desktop/my-3pi/src$ 
```

ToolsPrintSolution:

This code prints “on” every time the LED_PIN is set to true and “off” when the LED_PIN is set to false.



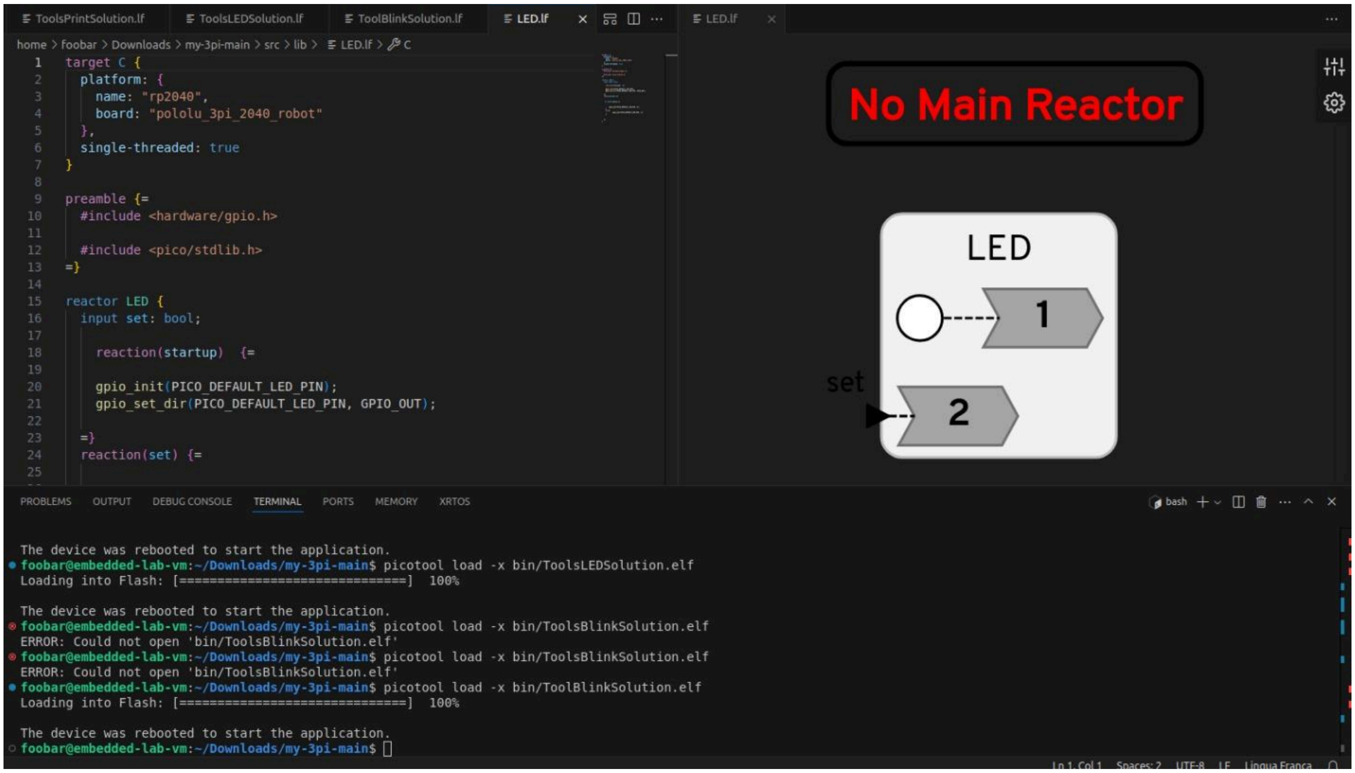
```
/**
 * Blink the default LED on the
 * <a href="https://www.pololu.com/docs/0J86">Pololu 3pi+ 2040 robot</
 * @author Abhi Gundrala
 * @author Edward A. Lee
 */
target C {
  platform: {
    name: "rp2040",
    board: "pololu_3pi_2040_robot"
  },
  single-threaded: true
}

preamble {=
  #include <stdio.h>

  #include <pico/stdlib.h>
  #include <hardware/gpio.h>
=}

main reactor {

  timer t(0, 400 ms)
  state led_on: bool = false
```

```
Off
On
Off
On
Off
On
Off
On
Off
On
Off
On
Off
On
```

LED.lf:

For the LED.lf code an input set is created of datatype boolean which will ensure true and false are allocated,

the first reaction startup is created it is initialized as soon as the program starts which initializes the GPIO pin for the led output. The reaction set input is accepted from the second code and based on its value the LED will turn on or off

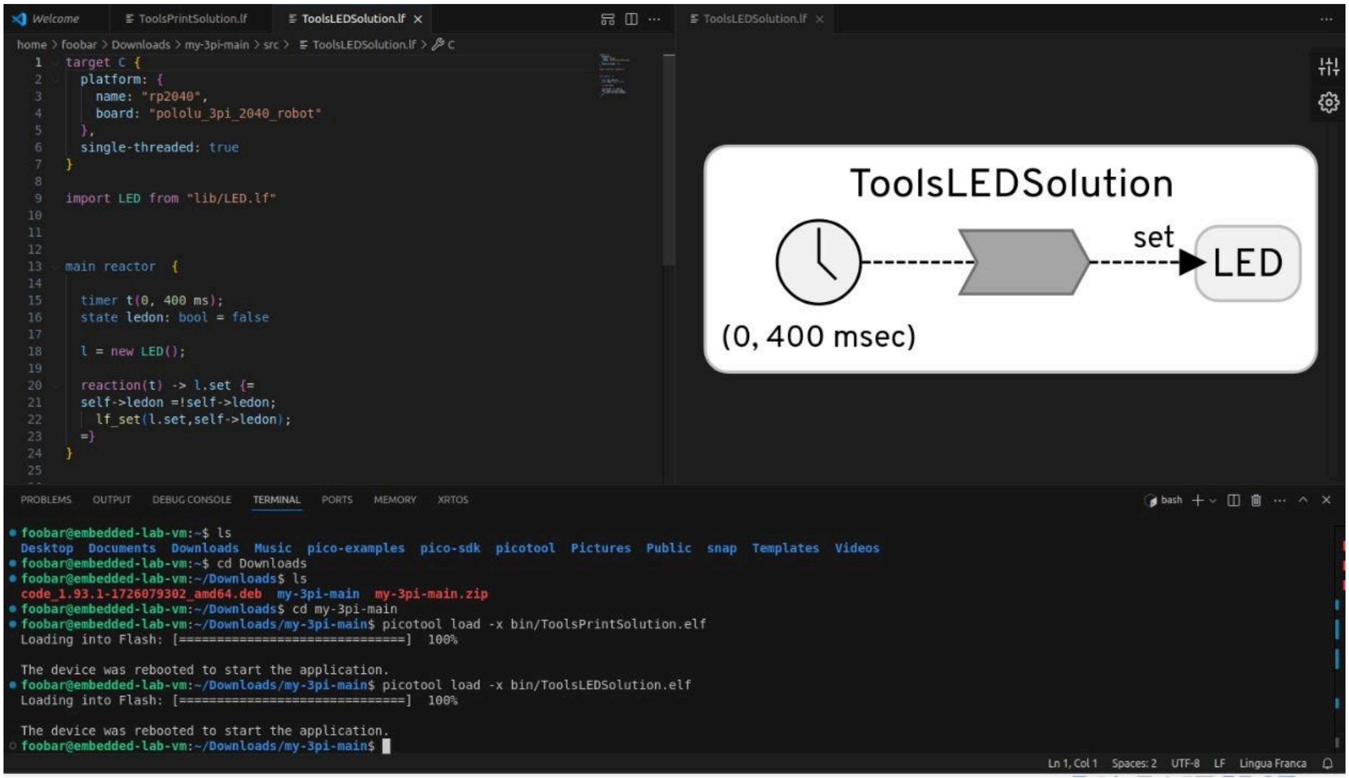


```
target C {
  platform: {
    name: "rp2040",
    board: "pololu_3pi_2040_robot"
  },
  single-threaded: true
}

preamble {=
  #include <hardware/gpio.h>

  #include <pico/stdlib.h>
=}

reactor LED {
  input set: bool;

    reaction(startup)  {=

    gpio_init(PICO_DEFAULT_LED_PIN);
    gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);

  =}
  reaction(set) {=

```

```
The device was rebooted to start the application.
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ picotool load -x bin/ToolsLEDSolution.elf
Loading into Flash: [==============================]  100%

The device was rebooted to start the application.
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ picotool load -x bin/ToolsBlinkSolution.elf
ERROR: Could not open 'bin/ToolsBlinkSolution.elf'
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ picotool load -x bin/ToolsBlinkSolution.elf
ERROR: Could not open 'bin/ToolsBlinkSolution.elf'
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ picotool load -x bin/ToolBlinkSolution.elf
Loading into Flash: [==============================]  100%

The device was rebooted to start the application.
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ 
```

ToolsLEDSolution.lf

The library was imported to the code, and a timer was introduced which triggers every 400milli seconds.A ledon state was made to be on. The imported LED.lf file is given to a variable l. In the reaction block which is called every 400 ms the value of ledon which is a boolean is changed, ie it will be True and False every 400 ms and since if the l.set value is changed the led will be changed based on the conditions mentioned in LED.lf



```
target C {
  platform: {
    name: "rp2040",
    board: "pololu_3pi_2040_robot"
  },
  single-threaded: true
}

import LED from "lib/LED.lf"



main reactor  {

  timer t(0, 400 ms);
  state ledon: bool = false

  l = new LED();

  reaction(t) -> l.set {=
  self->ledon =!self->ledon;
    lf_set(l.set,self->ledon);
  =}
}

```

```
foobar@embedded-lab-vm:~$ ls
Desktop  Documents  Downloads  Music  pico-examples  pico-sdk  picotool  Pictures  Public  snap  Templates  Videos
foobar@embedded-lab-vm:~$ cd Downloads
foobar@embedded-lab-vm:~/Downloads$ ls
code_1.93.1-1726079302_amd64.deb  my-3pi-main  my-3pi-main.zip
foobar@embedded-lab-vm:~/Downloads$ cd my-3pi-main
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ picotool load -x bin/ToolsPrintSolution.elf
Loading into Flash: [==============================]  100%

The device was rebooted to start the application.
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ picotool load -x bin/ToolsLEDSolution.elf
Loading into Flash: [==============================]  100%

The device was rebooted to start the application.
foobar@embedded-lab-vm:~/Downloads/my-3pi-main$ 
```

3. Postlab

3.1. What format specifier(s) for printf allows the printing of floats (there may be several)?

⇒ %f printing floating point , %e float in scientific notation or %E, %g or %G selects float or scientific notation automatically .

3.2. When might printf statements be the best tool for debugging?

⇒Printf acts like a checkpoint if the code is able to progress to the desired location for instance printf function inside a conditional if ,elseif statement would ensure if it has gotten within a particular elseif condition .

3.3. What other tools might be useful for debugging embedded software (note that using an interactive debugger like gdb with the robot or Pico board is possible but requires extra hardware)?

⇒ UART: use printf to send messages

Watchdog Timer: to detect system lockups and create a reset log to capture the state just before the reset.

3.4. What does the volatile keyword mean in C and when might you use it?

⇒Volatile keyword ensures the system always takes the value from the current inputs or states rather than using values available in the cache . Volatile is used when there is high likelihood of change in the status of a variable

3.5. What were your takeaways from the lab? What did you learn during the lab? Did any results in the lab surprise you?

⇒we learned how to use the proper usage of timer . library creation, also the thing that surprised us is that timers could overlap and scheduling is needed to avoid conflicts.