Pseudocode Document

Washing Machine Bros 2.0
Team ID: TW-LAM-009
Contraption ID: LR2C1

October 2024

1 Overview and Organization of this Document

This document is organized into three main sections to align closely with the original codebase, enhancing clarity and ease of understanding. The first section details the primary driver code, while the second outlines various library functions we developed, which are invoked by the primary driver code. Lastly, the third section provides a brief overview of the organization of our GitHub repository, accessible at LAMcompete.

2 Primary Driver Code

Algorithm 1 Embedded System Control Pseudocode

```
Importing Libraries:
From machine import necessary functions (e.g., enable_irq, disable_irq, PWM, I2C, Pin)
import time module

Initialize Variables:
total_weight ← 0
c ← 0                                                        ▷ Calibration Offset

Initialize Display:
Init_Display()
Clear_Display()

Initializing Pinch Valves:
Close_tube(Red)
Close_tube(Blue)

Calibrate Load Cell:                                         ▷ One-time calibration
c ← Tare()                                                   ▷ Tare the scale to set the offset
weights ← []                                                 ▷ List to store measured weights
for $i = 1$ to 10 do
    raw_wt ← Read() × 0.001
    weights.append(raw_wt)
    time.sleep_ms(1)
end for
weight ← ((sum(weights) / len(weights)) - c) × -0.517        ▷ Multiplying by the scaling offset
Print ”Average Weight:”, weight                              ▷ For reference on terminal

Start Process:
for cycle = 1 to 5 do                                        ▷ 5 cycles
    Open_Tube(pwmRED)
    Forward(100)
    time.sleep(2)                                            ▷ Push solution forward for 2s to make process faster
    while weight < 5 do
        Forward(60)
        weights ← []                                         ▷ List to store measured weights
        for $i = 1$ to 10 do
            raw_wt ← Read() × 0.001
            weights.append(raw_wt)
            time.sleep_ms(1)
        end for
        Write_on_Disp(”WASHING MACHINE BROS. 2.0 WEIGHT:”, weight)
        Show()
    end while

    Backwards(100)                                           ▷ Running peristaltic pump in reverse
    time.sleep(25)
    Backwards(40)                                            ▷ at different speeds
    time.sleep(10)
    Backwards(100)                                           ▷ to avoid cross-contamination
    time.sleep(25)
    Stop()
    Close_Tube(pwmRED)                                       ▷ Change pinch valve configuration for the Blue cup
    Open_Tube(pwmBLUE)
    total_weight += weight                                   ▷ Update total_weight
```

Algorithm 2 Embedded System Control Pseudocode (Continued)

```
    Re-calibrate Load Cell:
    c ← Tare()
    weights ← []
    for $i = 1$ to 10 do
        raw_wt ← Read() × 0.001
        weights.append(raw_wt)
        time.sleep_ms(0.001)
    end for
    weight ← ((sum(weights) / len(weights)) - c) × -0.517
    Print ”Average Weight:”, weight

    Rotate Stepper to Blue Cup:
    Rotate_Stepper_360(4, 2)                ▷ Rotate stepper by 4 rotations with 2ms delay
    Forward(100)
    time.sleep(3)                           ▷ Push solution forward for 3s to make process faster
    while weight < 10 do
        Forward(60)
        for $i = 1$ to 10 do
            raw_wt ← Read() × 0.001
            weights.append(raw_wt)
            time.sleep_ms(1)
        end for
        weight ← ((sum(weights) / len(weights)) - c) × -0.517
        Write_on_Disp(”WASHING MACHINE BROS. 2.0 WEIGHT:”, weight)
        Show()
    end while

    Backwards(100)                          ▷ Running peristaltic pump in reverse
    time.sleep(25)
    Backwards(40)                           ▷ at different speeds
    time.sleep(10)
    Backwards(100)                          ▷ to avoid cross-contamination
    time.sleep(25)
    Stop()
    Close_Tube(pwmBLUE)                     ▷ Closing Blue tube

    Rotate Stepper to Red Cup:
    Rotate_Stepper_Reverse(4, 2)            ▷ Rotate stepper by 4 rotations with 2ms delay
    total_weight += weight
    c ← Tare()
    weights ← []
    for $i = 1$ to 10 do
        raw_wt ← Read() × 0.001
        weights.append(raw_wt)
        time.sleep_ms(1)
    end for
    weight ← ((sum(weights) / len(weights)) - c) × -0.517      ▷ Multiplying by the scaling offset
    Print ”Average Weight:”, weight                             ▷ For reference on terminal
    Write_on_Disp(”WASHING MACHINE BROS. 2.0 WEIGHT:”, weight)
    Show()
end for
```

Algorithm 3 Import Necessary Modules

```
Import necessary modules:
From machine import necessary functions (e.g., enable_irq, disable_irq, PWM, I2C, Pin)
Import time module
```

Algorithm 4 Setup Hardware Pins

```
Setup all Hardware Pins:
pins for load cell:
clock_pin = Pin(18, Pin.out)
dat_pin = Pin(19, Pin.in)
Define pwmRED and pwmBLUE for pinch valves
Define IN1, IN2, IN3, IN4 for stepper motor
Define i2c for OLED screen
Define pin1, pin2, and enable for peristaltic pump
```

Algorithm 5 Load Cell Functions

```
Load Cell Functions:
Initialize constants: GAIN, OFFSET, SCALE, TIME_CONSTANT, FILTERED
function READ
    Disable interrupt for data_pin
    Check HX711 readiness
    Shift in data from HX711
    return resulting value
end function
function TARE(times)
    Measure and return average weight
end function
```

Algorithm 6 Pinch Valve Functions

```
Pinch Valve Functions:
Set PWM frequency for pinch valves
function CLOSE_TUBE(pwm)
    Set pwm to center position
end function
function OPEN_TUBE(pwm)
    Set pwm to 90-degree angle
end function
```

Algorithm 7 Stepper Motor Functions

```
Stepper Motor Functions:
Define full_step_sequence array
Define steps_per_revolution constant
function SET_STEP(p1, p2, p3, p4)
    Set motor pins IN1, IN2, IN3, IN4
end function
function ROTATE_STEPPER_360(revolutions, delay_ms)
    Rotate motor forward by full steps
end function
function ROTATE_STEPPER_REVERSE(revolutions, delay_ms)
    Rotate motor backward by full steps
end function
```

Algorithm 8 OLED Screen Functions

```
OLED Screen Functions:
Define constants for OLED commands (e.g., SET_CONTRAST, SET_DISP)
Define WIDTH, HEIGHT, ADDR for OLED display
Initialize display buffer
function WRITE_CMD(cmd)
    Send command to OLED
end function
function WRITE_DATA(data)
    Send data to OLED
end function
function INIT_DISPLAY
    Initialize display with predefined commands
end function
function CLEAR_DISPLAY
    Clear the display buffer
end function
function SET_PIXEL(x, y, color)
    Set pixel color in buffer
end function
function DRAW_CHAR(x, y, char)
    Draw character on OLED using font data
end function
```

3 Library Functions

Algorithm 9 Detailed Embedded System Control Algorithm

```
Import Libraries:
From machine import necessary functions (e.g., enable_irq, disable_irq, PWM, I2C, Pin)
import time module

Define Pin Numbers:
# For Load Cell:
Clock_Pin ← Pin 18 (output)                                  ▷ Load Cell Clock Pin
Data_Pin ← Pin 19 (input)                                    ▷ Load Cell Data Pin

# For Pinch Valves
PWM_RED ← PWM(Pin 15, mode=Pin.OUT)                          ▷ Pinch Valve - RED
PWM_BLUE ← PWM(Pin 16, mode=Pin.OUT)

# For Stepper Motor
IN1, IN2, IN3, IN4 ← Pins 28, 27, 26, 22 (output)           ▷ Stepper Motor Pins

# For OLED Screen
I2C_SCL ← Pin 5; I2C_SDA ← Pin 4                             ▷ I2C Pins for OLED

# For Peristaltic Pump (Define PWM signal)
Frequency ← 1000
Pin1 ← Pin 3 (output), Pin2 ← Pin 1 (output),
Enable ← PWM(Pin 2, Frequency)                               ▷ Peristaltic Pump Pins

Setup Variables:
# Setup Variables For Load Cell
Gain ← 1, Offset ← 0, Scale ← 1, Time_Constant ← 0.25, Filtered ← 0
# Setup Variables For Pinch Valve
pwmRED.freq(50)
pwmBLUE.freq(50)
# Setup Variables For Stepper Motor
full_step_sequence ← [[1, 1, 0, 0], [0, 1, 1, 0], [0, 0, 1, 1], [1, 0, 0, 1]]
steps_per_revolution ← 2048                 ▷ steps for full (360 degrees) rotation
# Setup variables for I2C OLED Screen
Defining Macros
WIDTH ← 128, HEIGHT ← 64, ADDR ← 0x3c                        ▷ OLED dimensions
# Setup Variables For Peristaltic Pump
min_duty ← 15000, max_duty ← 65535
```

Algorithm 10 Load Cell Functions

```
function READ_LOAD_CELL
    Disable IRQ for Data Pin
    for $i = 1$ to 500 do                                  ▷ Waiting until HX711 is ready
        if Data_Pin.value() = 0 then
            break
        end if
        time.sleep_ms(1)
    end for
    Result ← 0
    for $i = 1$ to $24 + \text{Gain}$ do                   ▷ Shift in data and gain & channel info
        Disable IRQ
        Clock_Pin ← HIGH, then LOW
        Enable IRQ
        Result ← (Result ≪ 1) OR Data_Pin.value()
    end for
    Result ≫= Gain                                         ▷ Shift back the extra bit
    if Result ¿ 0x7FFFFF then
        Result −= 0x1000000
    end if
    return Result
end function
function TARE_SCALE(times)
    Weights ← []
    for $i = 1$ to $times$ do                              ▷ Average over multiple measurements
        Raw_Wt ← Read_Load_Cell() ×0.001
        Weights.append(Raw_Wt)
        time.sleep(0.01)
    end for
    Avg_Weight ← sum(Weights) / len(Weights)
    return Avg_Weight
end function
```

Algorithm 11 Pinch Valve Functions

```
function CLOSE_TUBE(PWM)
    PWM.duty_u16(3276)                                     ▷ Set to center position
end function
function OPEN_TUBE(PWM)
    PWM.duty_u16(6553)                                     ▷ Set to 90-degree angle
end function
```

Algorithm 12 Stepper Motor Functions

```
function SET_STEP(p1, p2, p3, p4)
    IN1.value(p1)
    IN2.value(p2)
    IN3.value(p3)
    IN4.value(p4)
end function
function ROTATE_STEPPER_360(revolutions, delay_ms)
    for step = 1 to revolutions × steps_per_revolution do
        Current_Step ← step % length(full_step_sequence)
        Set_Step(full_step_sequence[Current_Step])
        time.sleep_ms(delay_ms)
    end for
    Set_Step(0, 0, 0, 0)
end function
function ROTATE_STEPPER_REVERSE(revolutions, delay_ms)
    for step = 1 to revolutions × steps_per_revolution do
        Current_Step ← step % length(full_step_sequence)
        Set_Step(full_step_sequence[-Current_Step - 1])
        time.sleep_ms(delay_ms)
    end for
    Set_Step(0, 0, 0, 0)
end function
```

Algorithm 13 OLED Screen Functions

```
function Init_Display
    Commands ← [SET_DISP — 0x00, SET_MEM_ADDR, 0x00, ..., SET_DISP — 0x01]
    for each cmd in Commands do
        Write_Cmd(cmd)
    end for
end function
function Clear_Display
    for $i = 1$ to length(Buffer) do
        Buffer[$i$] ← 0x00
    end for
end function
function Set_Pixel(x, y, color)
    if y > HEIGHT or x < 0 or y < 0 then
        return
    end if
    if x > WIDTH then
        y ← y + 8
    end if
    Page ← y // 8
    Bit ← y % 8
    Index ← Page × WIDTH + x
    if color then
        Buffer[Index] —= (1 ll Bit)
    else
        Buffer[Index] &= ∼(1 ll Bit)
    end if
end function
function Draw_Char(x, y, char)
    Font ← Dictionary of 5x8 font bitmaps
    for each line in Font[char] do
        for $j = 1$ to 5 do                              ▷ Draw each column of character
            Set_Pixel(x + j, y, line)
        end for
    end for
end function
```

Algorithm 14 Peristaltic Pump Functions

```
function DUTY_CYCLE(speed)
    if speed < 0 then
        speed ← 0
    end if
    if speed ¿ 100 then
        speed ← 100
    end if
    duty_cycle ← $min\_duty + ((max\_duty - min\_duty) \times (speed/100))$
    Enable.duty_u16(duty_cycle)
end function
function Run_Pump_For_Time(duration)
    Enable.duty_u16(0)                                   ▷ Stop Pump
    time.sleep(duration)
    Enable.duty_u16(0)
end function
```

4 Github Repository

The repository is organized into distinct folders for modularity, code reusability, and debugging efficiency. This structure allows testing of individual components without affecting the main codebase, simplifying future maintenance and updates.

1. **OLED**:
   - *i2c_scanner.py*: Scans I2C devices to detect the OLED screen's address.
   - *i2c_oled.py*: OLED library containing custom functions and instruction sets in 5x8 format for alphanumeric and special character display.
   - *test_lib.py*: Tests the custom OLED library.
2. **Peristaltic_pump**:
   - *peristaltic_pump.py*: Provides functions to control pump pressure and manage bi-directional water flow.
3. **Load_cell**:
   - *final-load_cell.py*: Contains functions to operate the load cell like taring and weighing.
   - *for_sf.py*: Used to find the scaling factor.
4. **Servo & Stepper**:
   - *servo.py*: Contains functions to open and close the pinch valve servo motors.
   - *stepper.py*: Manages functions for the linear actuator's stepper motor. To move the lead screw nut between the two cups.
5. **Integrated code**:
   - *Code_Lib.py*: Custom library defining pin configurations and key functions for all components.
   - *Code_run.py*: Executes the complete setup, following the sequence outlined in the system flowchart using the custom library, *Code_Lib.py.*
   - *Test_code.py*: Used for testing individual components independently with *Code_Lib.py*, avoiding interfering with the final python program.
6. **combine (ARCHIVE)**:
   - A deprecated folder originally used for combined component testing, now archived for reference.

This structured approach optimizes code readability, facilitates future enhancements, and ensures a stable testing environment for each system component before full integration.