OIL FURNACE CONTROLLER

Design Laboratory Project

Prepared by: Jakub Kłoczko

Piotr Klimala

1. What’s the problem?

Self-made oil furnaces have a lot of components that need to be controlled such as oil pump, fan, heater, igniter, and flame sensor. There are many market positions: the original Siemens controllers, and some Chinese equivalent with obviously better price. None of these proposals has a heater and fire control integrated into one sophisticated system. The user can’t manipulate the pre-purge time, and the amount of retakes when the flame goes down. We also can’t control the energy use of a furnace, oil use and estimate the whole time when the furnace was on or heater working time.

2. Operating Environment of Oil Furnace Controllers

- **Temperature Variations**
    - Range: Controllers must reliably function across a wide temperature range, as they are used in various climates.
    - Impact: They should withstand sudden temperature changes without performance degradation, ensuring consistent operation.
- **Humidity and Moisture**
    - Exposure: Presence of humidity and potential moisture, especially in basements or utility rooms where furnaces are often located.
    - Protection: Controllers should be designed to protect against moisture, preventing corrosion or electrical issues.
- **Dust**
    - Environment: Furnace rooms can be dusty, necessitating some level of dust resistance in controllers.
    - Maintenance: The design should minimize the impact of dust and allow for easy cleaning.
- **Mechanical Vibrations**
    - Sources: Operating furnaces can produce vibrations that may affect electronic components.
    - Design Consideration: Robust construction and vibration dampening are important to maintain accuracy and longevity.
- **Accessibility and User Interaction**
    - Location: Controllers are often placed in less accessible areas, like basements or utility closets.
    - Design for User: Interfaces should be simple and easy to use, considering infrequent interactions or challenging conditions.
- **Connectivity and Remote Access**
    - Network Environment: For smart controllers, stable connectivity is essential for remote monitoring and control.
    - Consideration: Reliable wireless or wired networking capabilities are crucial for consistent communication.

- **Safety and Compliance**
    - Regulations: Controllers must comply with safety standards and regulations for heating systems and electronic devices.
    - Safety Features: Incorporation of safety features to prevent hazards like overheating or electrical malfunctions is critical

3. Advantages of our solution (design assumptions)

- **Integration of Advanced Control Functions**
    - The Cortex-M0 allows for the integration of advanced control algorithms. These include precise temperature control, adaptive algorithms, and diagnostic functions. This enhances the overall efficiency and reliability of the system, improving the heating performance and safety.
- **Flexibility and Scalability**
    - Due to its flexibility, the Cortex-M0 enables easy adaptation of the controller to different models of oil furnaces. Its scalability also facilitates easy software updates, key for keeping the controller up-to-date with new technologies and standards.
- **Cost-Effectiveness**
    - Being part of the widely accessible ARM platform, the Cortex-M0 may be more cost-effective compared to specialized, niche microprocessors used in traditional controllers. This cost reduction can translate into a lower final product price without compromising on quality or functionality.
- **Reduced System Complexity**
    - Utilizing the Cortex-M0 can lead to a simplified system architecture, which in turn can reduce the likelihood of failures and facilitate maintenance. A simpler system often also means easier operation for end-users.
- **Support for Wireless Connectivity and IoT**
    - The Cortex-M0 is compatible with modern wireless and IoT (Internet of Things) technologies. This allows for remote monitoring and control of the oil furnace, offering new possibilities such as remote control and energy consumption optimization for users.
- **Simple Temperature Regulation**
    - Mechanical Thermostats: Most mechanical controllers use simple thermostats for temperature regulation. Users set the desired temperature using a knob or slider.
    - Response to Temperature Changes: These devices operate on the principle of thermal expansion of materials, activating or turning off the furnace when the temperature in the room reaches a certain level.
- **Limited Programming Features**
    - Simple Schedules: Some analog controllers may offer basic programming functions, such as different settings for day and night, but they are much less flexible than digital controllers.
- **Lack of Remote Control**

    - Manual Control: Mechanical controllers require manual adjustment and do not offer remote control or monitoring, which are typical for modern digital systems.
- **Reliability and Durability**
    - Less Complex: Due to their simplicity, analog controllers are often considered less prone to failure and easier to maintain than their digital counterparts.
    - Longer Lifespan: They typically have a longer lifespan due to the lack of complex electronic components.

- **Limited Diagnostics and Safety**
    - Basic Safety Features: Although they may include some basic safety features, such as an overheat switch, they usually do not offer advanced diagnostic features and alarms.
- **Cost and Availability**
    - Low Price: Analog controllers are usually cheaper to purchase than digital controllers.
- **Enhanced Functionality with C Language Code Base**
    - The project leverages the C programming language for its codebase, offering performance and efficiency benefits. This could provide an advantage over competitors using higher-level languages.
- **Basic Functionality Implementation**
    - Key basic functionalities, including a robust starting procedure and sophisticated error handling for flame failures, may offer improved safety and reliability over standard controllers.
- **Remote Control via API Communication**
    - The inclusion of an API for remote control positions the project as a modern solution, potentially outperforming traditional systems that lack such connectivity.
- **User Interface Considerations**
    - The addition of a screen for displaying basic parameters and enabling manual adjustments via buttons enhances the user interface, likely surpassing more basic interaction methods found in competing models.
- **Advanced PCB Design with Cortex M0 Microprocessor**
    - Employing a PCB with the Cortex M0 microprocessor offers advantages in processing power and energy efficiency, distinguishing the project as a forward-thinking solution.

4. Solutions from market

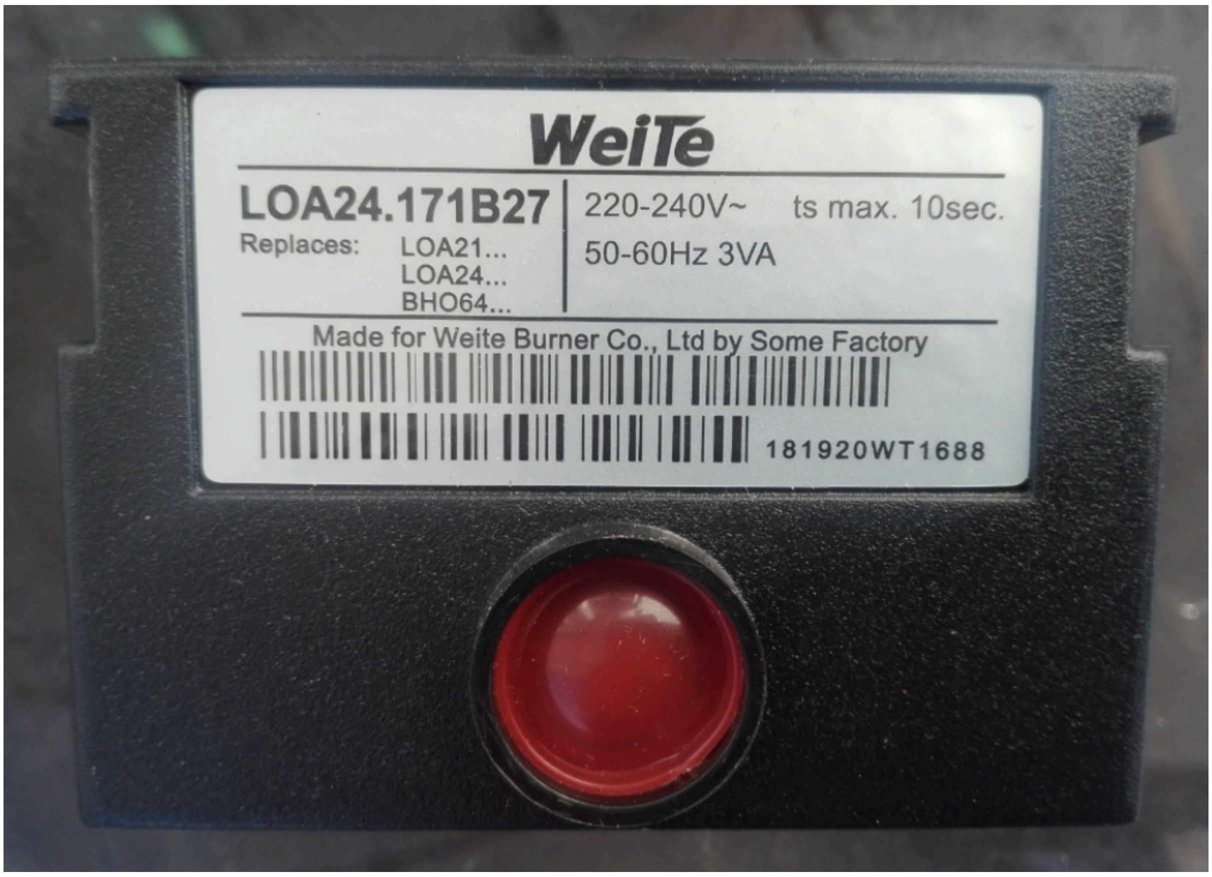
There are a lot of commonly used devices but everyone have the same problem - sophisticated mechanical design which occur a high price and is terribly hard to repair.

- **LOA24 siemens**

Very popular controller, but in very high price (over 500zł) non customizable (if we change the furnace = we need to change controller), need a outside heater controller, nice attachment to make tests, no data output to remotely checking the parameters

- **LOA24 Weite**

Not reliable as siemens but in very good price so widely used by amators in private houses; like siemens - no internet connection (actually any connection or data output), no test adapters, generally - very good copy of above one


WeiTe
LOA24.171B27
Replaces: LOA21... LOA24... BHO64...
220-240V~ ts max. 10sec.
50-60Hz 3VA
Made for Weite Burner Co., Ltd by Some Factory
181920WT1688

Both of them are in use, it depends from financial conditions of users and place where it will be mounted.

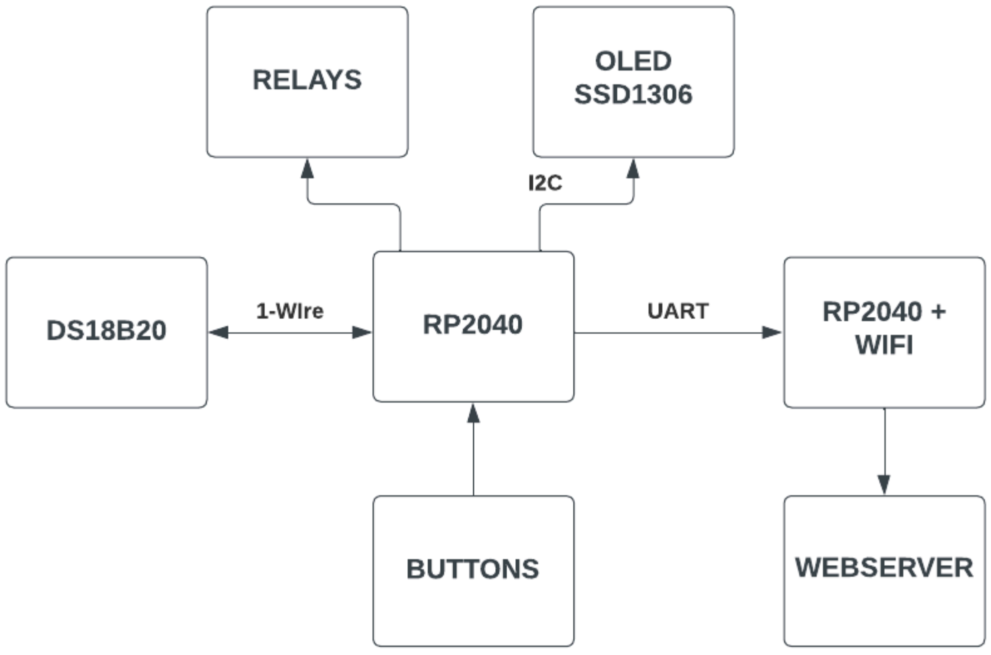
5. Idea

Using some state of the art technology we want to propose the device which is:

- **very cheap** (RP2040 processor, some relays, OLED screen, some buttons and simple PCB)
- **customizable** (user will have a opportunity to change every parameter like wanted oil temp, pre-purge and igniter time etc)
- **open to world** (device will be connected to global internet by wifi so user can check parameters and current status of device from every place in the world)
- **easy to repair** (simple PCB design is a highway to easy replace the wear elements)
- **upgradable** (every software upgrade can implement a new functions without replacing a hole device)

In comparison with currently market competition we will deliver such a new product. It probably will gain interest among a large group of people if we advertise our product as a solution for unsolved problems. For sure, it will have much better price than other products, have a internet connection which is one of the most important features of any smart home gadgets and can be used more seriously in professional conditions like factory.

6. PROJECT



Pic.1 block diagram

Furnace.py **structure:**

- class **Furnace:**
    - flame_detector_pin - flame detector (Pin 6, IN, PULL UP, GPIO)
    - valve_pin - valve (Pin 2, OUT, GPIO)
    - heater_pin - heater(Pin 3, OUT, GPIO)
    - fan_pin - fan(Pin 4, OUT, GPIO)
    - magneto_pin - ignition magneto(Pin 5 , OUT , GPIO)
    - pre_purge_time - constant [5s]
    - igniter_time - constant [6s]
    - curr_state - variable [1 at start]
    - oil_temp - variable [30 default]
    - hysteresis_width - variable [1 default]
    - error_count - variable [0 at start]
    - working_flag - flag [true when furnace starts]
    - oil_good - flag [true when temperature reached]
    - failure - flag [when true need hard reset]
    - temp_sensor - temperature sensor ds18b20 (Pin 7, 1-Wire)
    - temp - variable (checked temperature)
    - oil_watch - periodic timer (on whole time)
    - tim - one shot timer (used during states)
    - start_tim - periodic timer (on before start)
    - flame_watch - periodic timer (on after ignition)

      Integral methods:
        - check_temp - called when current oil temperature is need
        - oil_temp_check - called by oil_watch timer to check if is need to toggle heater
        - real_state_machine - called when is moment to start the fire
        - next_state - used to move through states
        - start_procedure - called by start_tim when furnace is preparing to start
        - set_oil_temp - called to change desire temperature
        - set_hysteresis_width - called to change desire hysteresis width

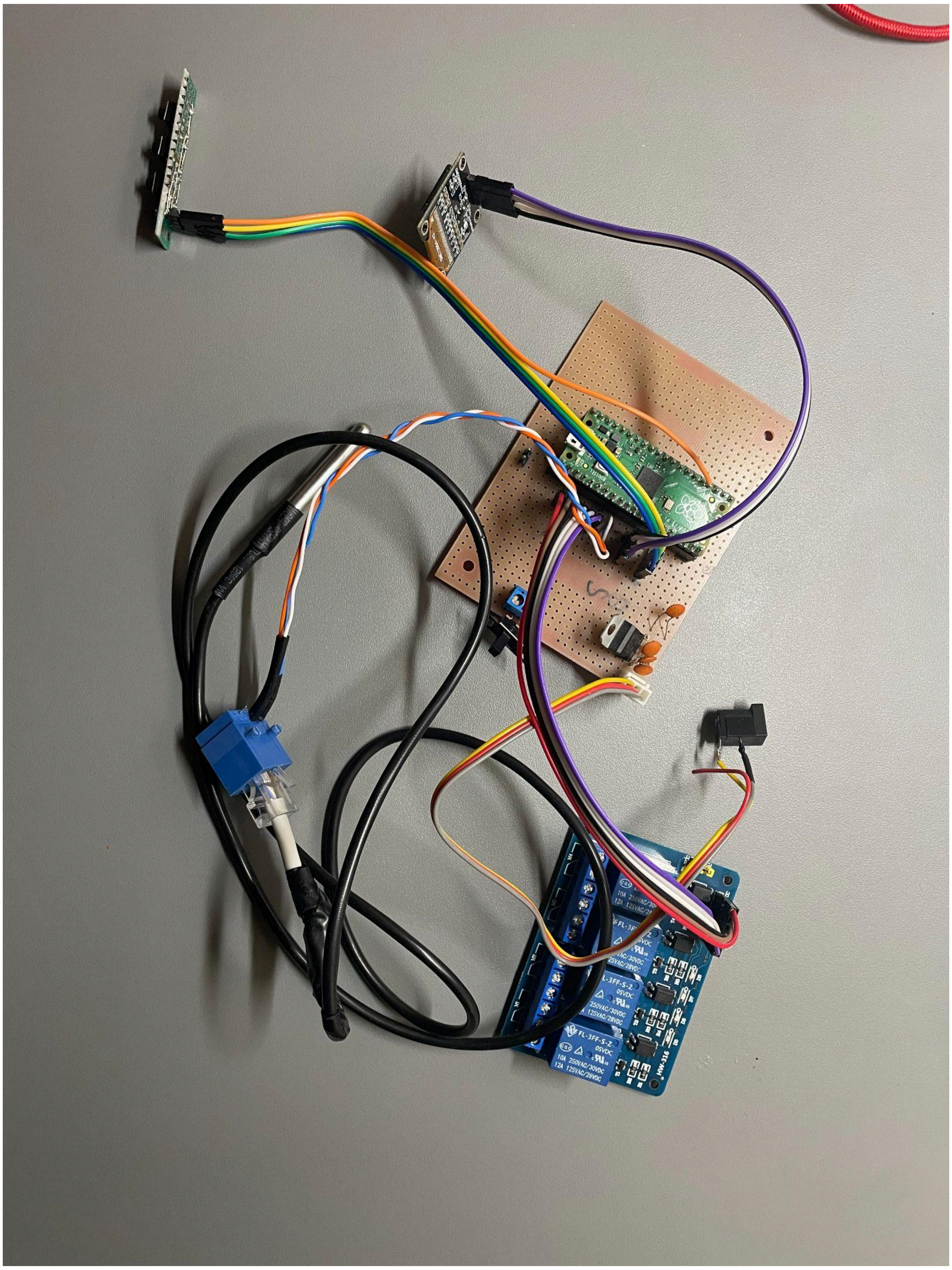
main.py **structure:**

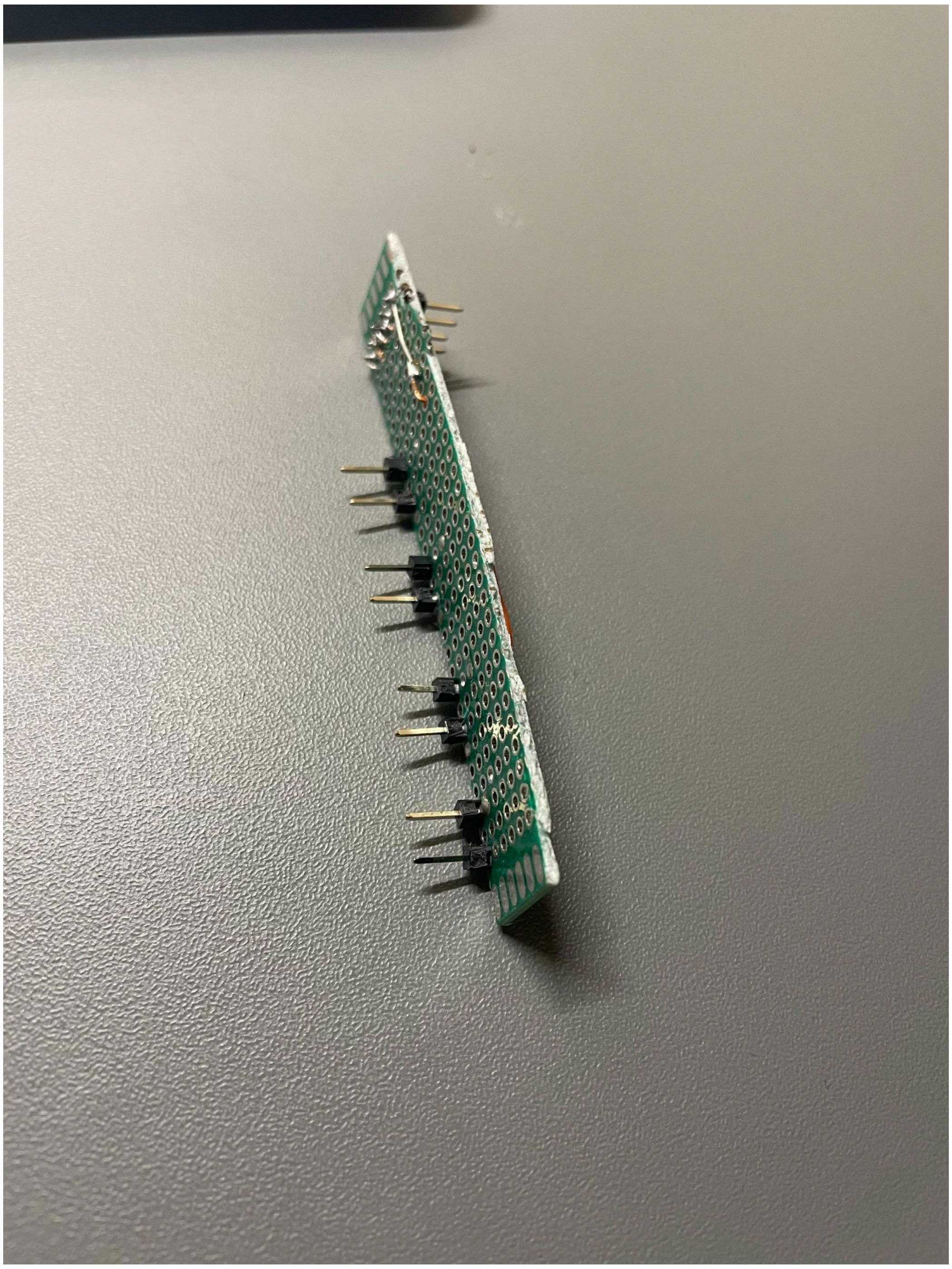
- alloc_emergency_exception_buf - needed when irq's are enable
- menu_flag - flag [true when in menu]
- display - ssd1306 oled display (Pin 8 SDA, 9 SCL , I2C)
- up_pin - Button up (Pin 10, IN , PULL UP, IRQ Enable - falling)
- down_pin - Button down (Pin 11, IN , PULL UP, IRQ Enable - falling)
- menu_pin - Menu/OK pin (Pin 18, IN , PULL UP, IRQ Enable - falling)

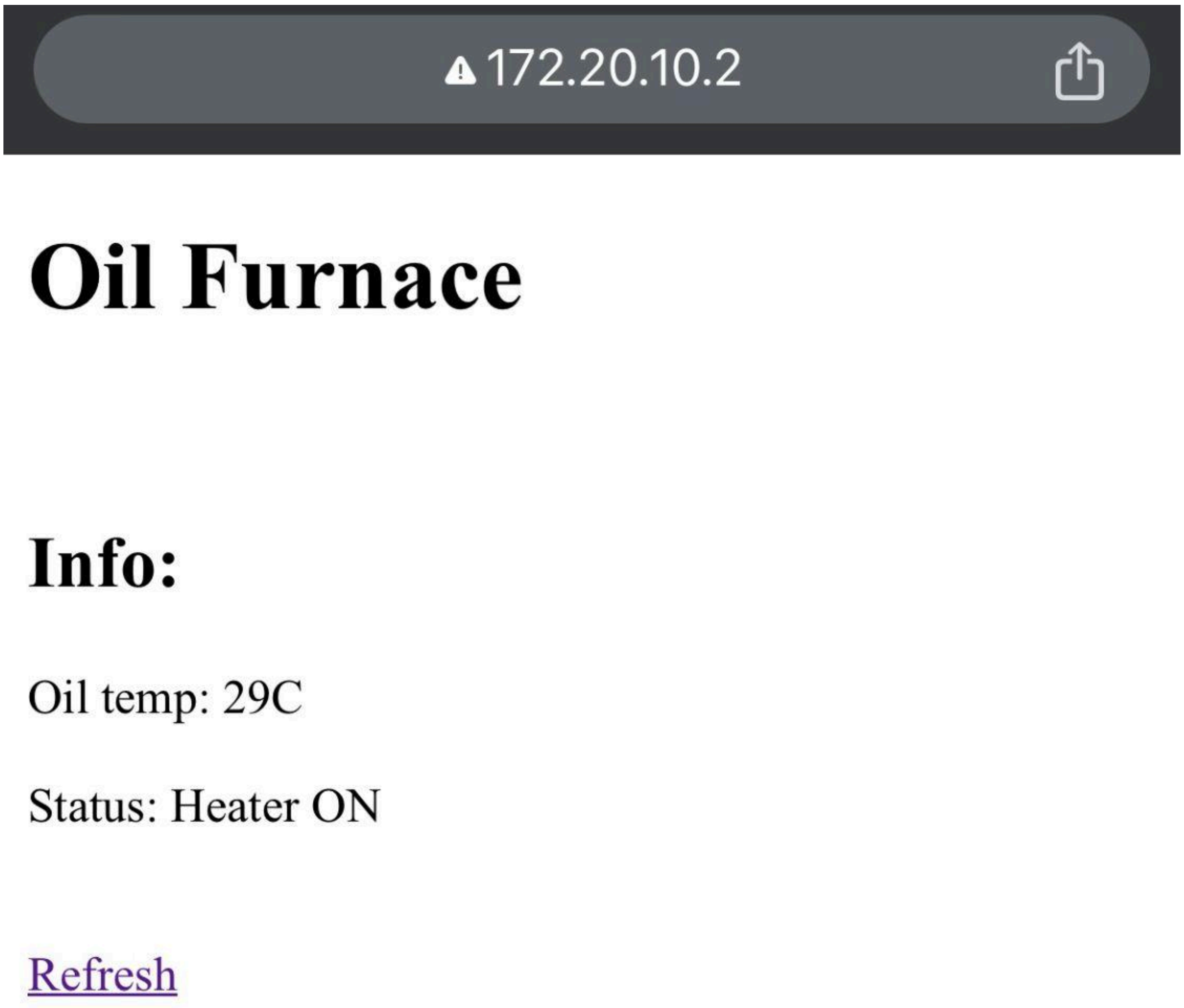
  Methods:
    - de_menu - toggle menu flag
    - menu_click - menu button handler, callback from up pin irq
    - menu_click - up button handler, callback from menu pin irq
    - menu_move_down - down button handler, callback from down pin irq

Full code for both pico is available on github:
https://github.com/Ramkomak/oil_furnace_control

7. Prototype


Pic. 2 full prepared design


pic. 3 adapter for timing tests


172.20.10.2

Oil Furnace

Info:

Oil temp: 29C

Status: Heater ON

Refresh

pic. 4 webpage with basic info