# Практична реалізація модулю

## 1. Апаратна частина

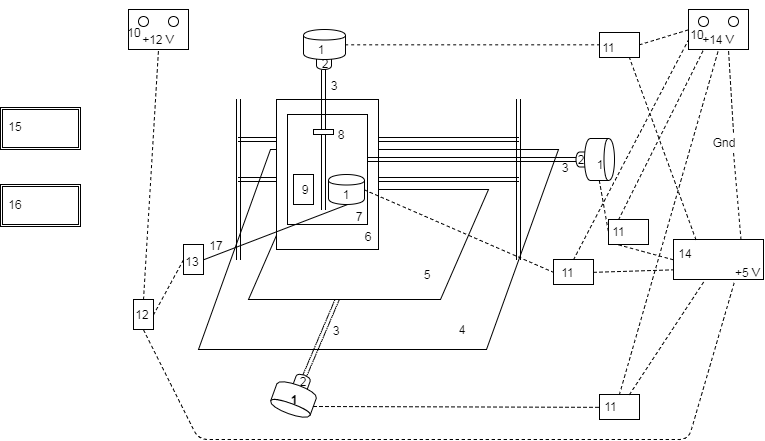
Апаратна частина модулю складається з осей від 3D принтеру, робочої площадки, вертикальної площадки, муфт, крокових двигунів та мікроконтролеру з камерою, насосу, реле, механічних кінцевиків (Endstop Switch Module) джерел живлення, драйверів крокових двигунів, керуючого мікроконтролеру, кріплень (гайок, болтів, ізоляційної стрічки тощо) та проводів.

Загальну схему модулю можна побачити на мал. 1. На даному малюнку 1 – крокові двигуни, 2 – муфти, 3 – осі, 4 – основа, 5 – робоча площадка, 6 – основа для кріплення осі OY, 7 – основа для кріплення осі OZ, 8 – тримач, 9 – модуль отримання зображення (Raspberry Pi та Raspberry Pi Camera Module), 10 – джерела живлення, 11 – драйвери крокових двигунів, 12 – реле, 13 – насос, 14 – керуючий мікроконтролер (Arduino Mega 2560), 15 – маршрутизатор для отримання зображення від Raspberry Pi, 16 – комп’ютер, 17 – трубка. Зовнішній вигляд апаратної частини модулю зображений на малюнку 2.

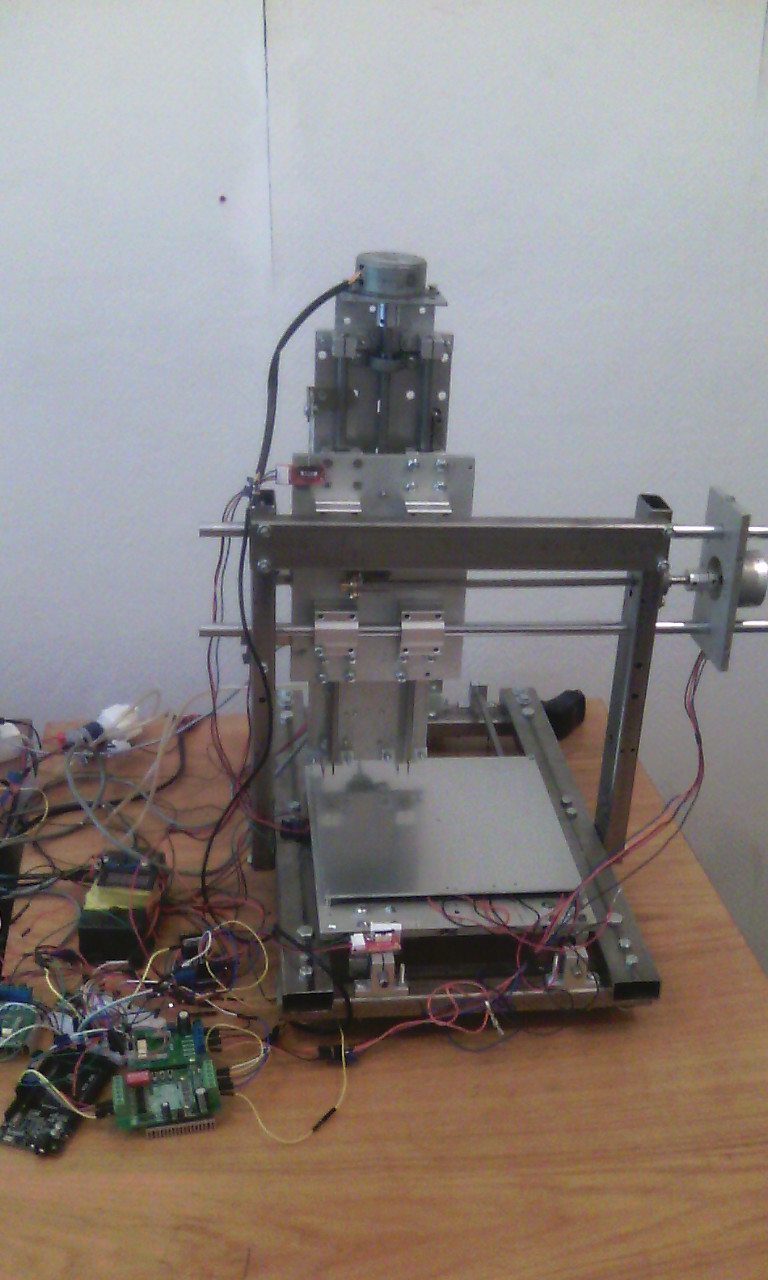
Габарити вертикальної площадки ≈ 39x11 см2.

Габарити горизонтальної площадки ≈ 26x22 см2.

Параметри деталей можуть варіюватися. Проте умовою є можливості насосу захопити деталь даної ваги та форми, а також темний колір деталі (аби вона краще розпізнавалась, адже деталі знаходяться на білому фоні). Проте і ці обмеження можна дещо пом’якшити – можна за необхідності замінити насос на більш потужний, зробити насадку на робочий носик зручнішої форми, світлі деталі розміщувати на темному фоні.



Мал. 1 – Загальна схема модулю



Мал. 2 – Вигляд апаратної частини модулю

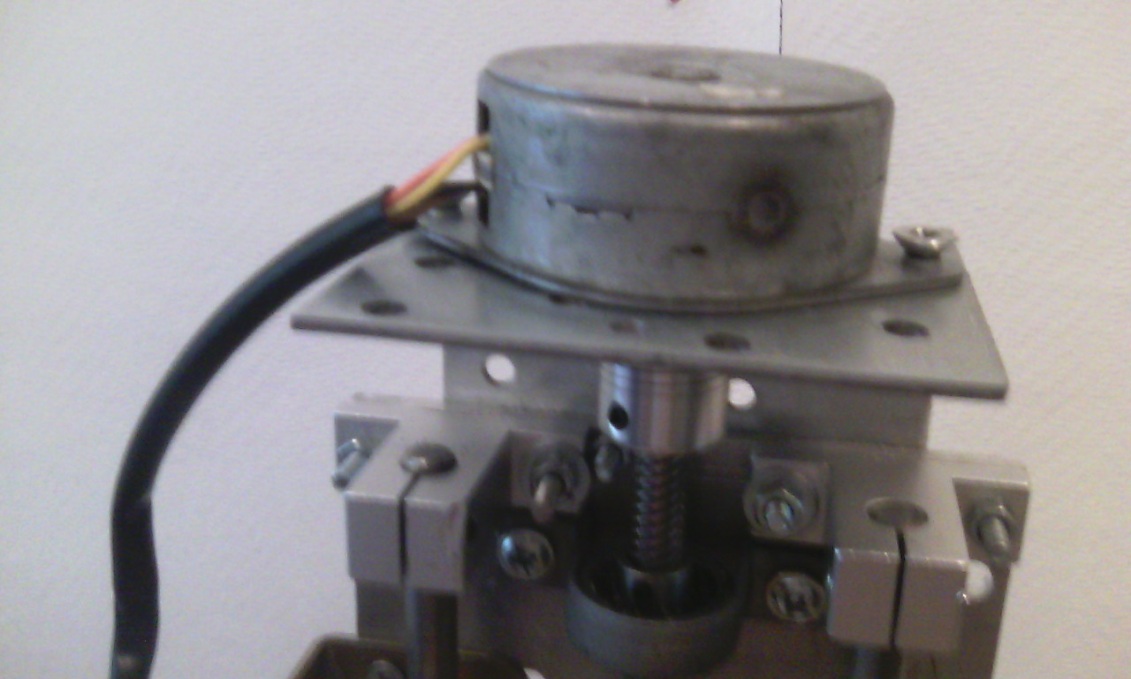
Послідовність роботи модулю наступна:

1. Мікроконтролер (мал. 1, елемент 14) подає сигнали для установлення всіх площадок в початкове положення. Це положення в якому площадки замикають всі механічні кінцевики і є точкою відліку (координати O(0,0,0)). Сигнали від мікроконтролеру подаються до драйверів крокових двигунів, які в свою чергу живлять двигуни від зовнішнього джерела. Площадки переміщуються кроковими двигунами, що прикріплені до гвинтових осей (мал. 1, елемент 3) за допомогою муфт (мал. 1, елемент 2).
2. Модуль з камерою, закріплений на вертикальній площадці ((мал. 1, елемент 9) з певним часовим інтервалом (напр. 2 секунди) захоплює зображення та передає його. В даній роботі використовувався Raspberry Pi та Raspberry Pi Camera Module. Raspberry Pi – одноплатний комп’ютер, що керується операційною системою. Тому для отримання зображень з інтервалом в 2 секунди був написаний відповідний bash-скрипт (див. додаток 1)//write it.
3. Зображення можна отримати на комп’ютер підключивши Raspberry Pi та комп’ютер до маршрутизатору(мал. 1, елементи 9, 16 та 15) та перейшовши за адресою 192.168.1.100:8080.
4. Зображення оброблюється оператором за необхідності та визначаються координати найближчого елемента.
5. Крокові двигуни повинні перемістити площадки в визначені координати, опустити вертикальну пощадку (мал. 1, елементи 7 і 8), подати дозволяючий сигнал на реле (мал. 1, елемент 12), що живить насос (мал. 1, елемент 13) від зовнішнього джерела (мал. 1, елемент 10), перемістити захоплену деталь, за необхідності позиціонувати її та вимкнути насос (відповідний сигнал реле від мікроконтролеру).
6. Повторити від пункту 2, доки не будуть розставлені всі необхідні деталі, або доки не закінчаться деталі з робочої площадки.

### 1.1 Переміщення по осям

Вертикальна площадка переміщається по осям OY та OZ. Площадка з елементами та платою (горизонтальна площадка) переміщається по OX. Переміщення здійснюється за допомогою крокових двигунів.

В даній роботі тип двигунів (уніполярні чи біполярні) не є критичною характеристикою модулю, адже уніполярний двигун можна використовувати як біполярний просто залишивши загальний провід непідключеним, обо підключити його до спільної землі. Для переміщення по осям використовувались 2 біполярні двигуни (мал. 3) та 1 уніполярний (бо ці двигуни повинні переміщати навантаження – площадки, а біполярні потужніші за аналогічні уніполярні двигуни).



Мал. 3 – Біполярний двигун на осі модулю

Сигнали для переміщення передаються мікроконтролером (в даній роботі використовувався Arduino Mega 2560 (мал. 1, елемент 14)) до драйверу відповідного двигуна (для переміщення по осям використовувались 2 драйвери A3977 та 1 драйвер TB6560 V2).

Драйвери відповідно до отриманих сигналів подають на двигуни послідовність сигналів з величиною напруги пропорційною напрузі живлення. Таким чином земля зовнішнього джерела живлення і земля мікроконтролеру повинна бути спільною. В деяких драйверах це вже реалізовано апаратно (зазвичай в них є спеціальні входи для 2-х сигналів землі – один для землі джерела живлення, один для землі мікроконтролеру), в деяких землі потрібно підключати разом (в таких драйверах зазвичай один вхід для землі).

Отримана від драйверів послідовність сигналів повертає вал двигуна. Дані двигуни прикріплені до гвинтових осей за допомогою муфт. Таким чином при повороті валу двигуна повертається відповідна гвинтова вісь, що в свою чергу переміщує відповідну площадку.

В залежності від характеристик крокового двигуна, тертя на осі тощо, можливе різне співвідношення параметрів кількість кроків двигуна – довжина пройденого шляху. Емпіричні вимірювання цих параметрів в даного модулю наведені в таблиці 1:

Таблиця 1 – Співвідношення параметрів кількість кроків двигуна – довжина пройденого шляху для різних осей модулюю

|  |  |  |
| --- | --- | --- |
| Вісь | Кількість кроків | Пройдена відстань (мм) |
| OX | 100 | 3 |
| OY | 100 | 1.5 |
| OZ | 100 | 0.85 |

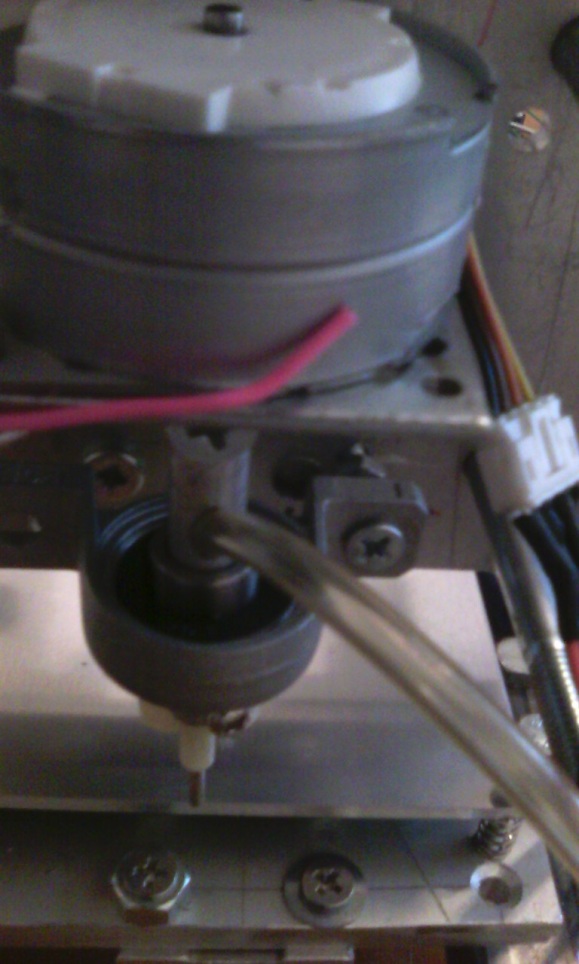
Проте ці дані не є точним орієнтиром, адже іноді крокові двигуни пропускають кроки. Саме тому необхідний зворотній зв’язок для контролю положення площадок модулю. Цей зворотній зв’язок здійснюється за допомогою камери. Зображення з камери бажано отримувати і обробляти в реальному часі, але для камери Raspberry Pi Camera Module, яка була використана в даній роботі, існує технологічне обмеження – камера може захватувати зображення 1 раз в 2 секунди. Цей час потрібен камері аби встановити її рівні освітленості.

### 1.2 Захоплення та позиціонування деталі

Коли робочий носик знаходиться над деталлю це фіксується камерою. Відповідне зображення через маршрутизатор передається до комп’ютеру. Дана ситуація повинна бути зафіксована обробляючою програмою (різка зміна координат найближчої деталі) і керуючому мікроконтролеру повинен бути переданий відповідний сигнал (наприклад по UART(universal asynchronous receiver/transmitter)). Обробка даного сигналу повинна здійснюватись контролером у перериванні для оперативного реагування. Мікроконтролер у відповідь повинен подати послідовність сигналів до драйверу крокового двигуна що керує переміщенням по осі OZ та опустити кроковий двигун з насадкою (робочим носиком), що прикріплені до вертикальної площадки. Після цього необхідно подати дозволяючий сигнал на реле, що в свою чергу ввімкне живлення насосу від зовнішнього джерела.

Насос трубкою з’єднаний з кроковим двигуном що призначений для позиціонування деталі. Таким чином насос повинен захватити деталь через носик двигуна. Далі мікроконтролер повинен подати сигнали драйверу крокового двигуна що керує переміщенням по осі OZ та підняти кроковий двигун з насадкою (піднімання та опускання вертикальної площадки повинне здійснюватись на невелику, але фіксовану величину). Деталь потрібно перемістити на необхідну позицію. Під час переміщення насос повинен бути ввімкнутим (інакше деталь просто випаде).

Перемістившись до відповідної координати, деталь, за необхідності, позиціонують. В даному випадку позиціонування може здійснюватись поворотом відповідного крокового двигуна. Для позиціонування використовувався біполярний двигун (мал. 4) та драйвер L298N.



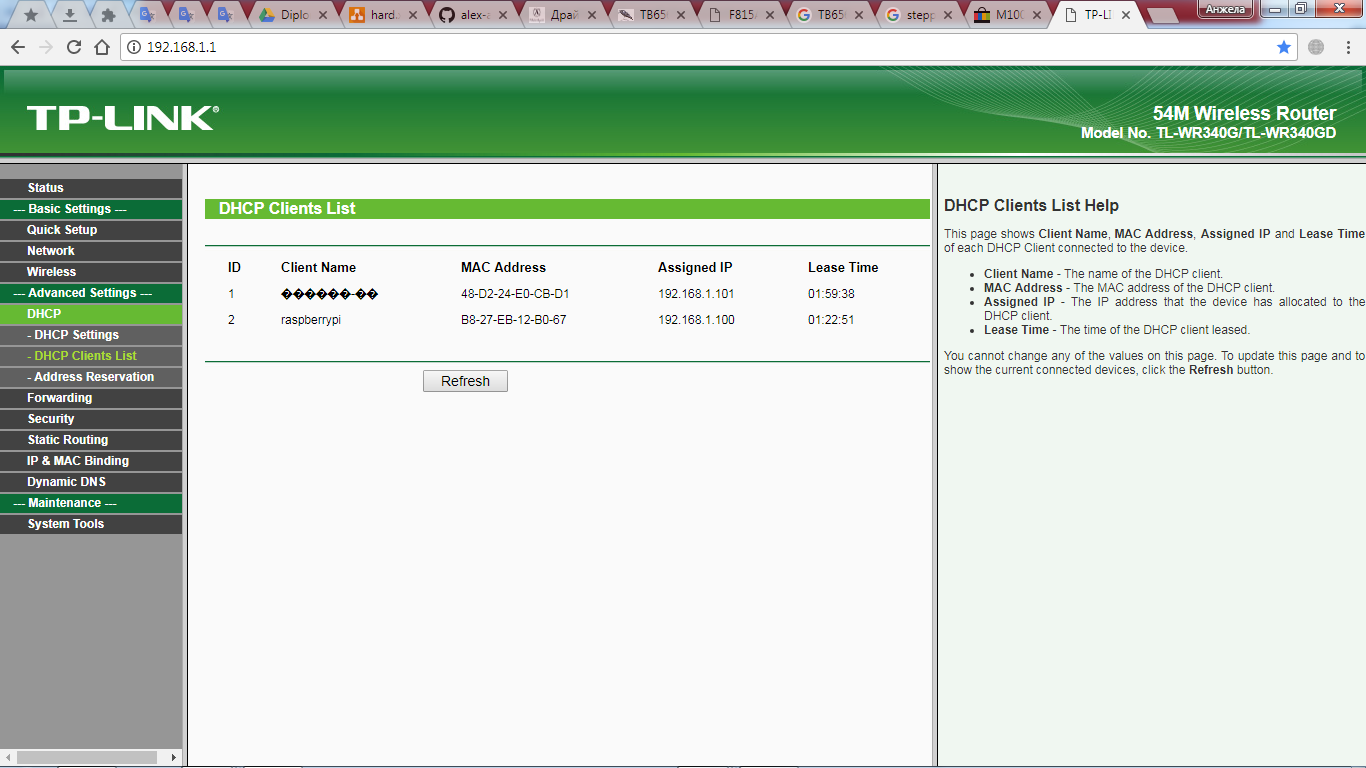
Мал. 4 – Біполярний кроковий двигун для позиціонування деталі

Мікроконтролер повинен подати сигнали драйверу крокового двигуна що керує переміщенням по осі OZ та опустити кроковий двигун з насадкою і подати сигнал реле для відключення насосу. Деталь залишиться на місці. Мікроконтролер повинен подати сигнали драйверу крокового двигуна що керує переміщенням по осі OZ та підняти кроковий двигун з насадкою. Далі необхідно знову здійснити пошук найближчої деталі та почати все спочатку доки не закінчаться деталі, або доки не будуть встановлені всі необхідні деталі.

## 2. Програмна частина

### 2.1 Захоплення зображення з камери

В даній роботі використовувалися Raspberry Pi та Raspberry Pi Camera Module. Raspberry Pi – одноплатний комп’ютер, що керується операційною системою. Була установлена Raspbian GNU/Linux 7 (wheezy). Для доступу до Raspberry Pi використовувався маршрутизатор: Raspberry та комп’ютер підключались до маршрутизатору. Далі визначалась IP адреса Raspberry (мал. 5)



Мал. 5 – Визначення IP адреси Raspberry Pi

Для доступу за цією адресою використовувалась програма PuTTY. Для того аби на Raspberry Pi 1 раз в 2 секунди захоплювалось зображення, був написаний bash-скрипт (додаток 1//write it). Зображення з камери бажано отримувати і обробляти в реальному часі, але для камери Raspberry Pi Camera Module існує технологічне обмеження – камера може захватувати зображення максимум 1 раз в 2 секунди. Цей час потрібен камері аби встановити її рівні освітленості. Приклад захопленого зображення – мал.6.



Мал. 6 – Зображення робочої площадки отримане з камери

Доступ до даного зображення можливий за адресою < Raspberry\_Pi\_IP\_address>:8080.

### 2.2 Обробка зображення та розпізнавання деталей

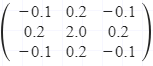
Була використана бібліотек OpenCV. Лістинг – додаток 2 //write it. Розпізнавалися темні деталі на світлому фоні. Для розпізнавання деталей використовувався детектор кордонів Кенні.

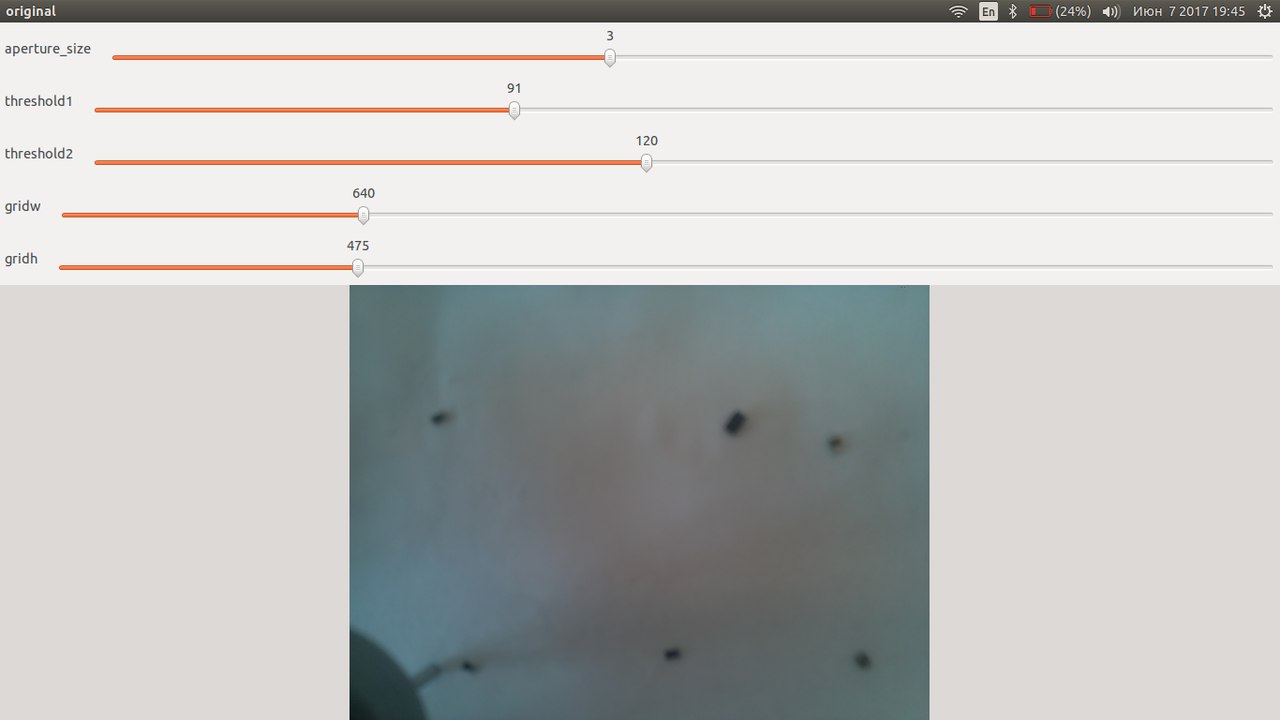
Спочатку в зображення збільшується яскравість для зменшення впливу шумів та можливого недостатнього освітлення. В результаті отримуємо картинку зображену на малюнку 7.



Мал. 7 – Картинка після застосування початкової згортки

Це здійснюється за допомогою згортки з матрицею:





Мал. 8 – Інтерфейс програми

Інтерфейс даної програми можна побачити на мал. 8. Параметр aperture\_size відповідає за розмір оператора Собеля (див. розділ … пункт …//fin ver). aperture\_size може приймати значення 1, 3, 5 або 7. Проте в програмі можна виставити значення цього параметру 3, 5 або 7:

if(pos>=3){

if(pos%2==1)

Hmin = pos;

else Hmin = pos+1;

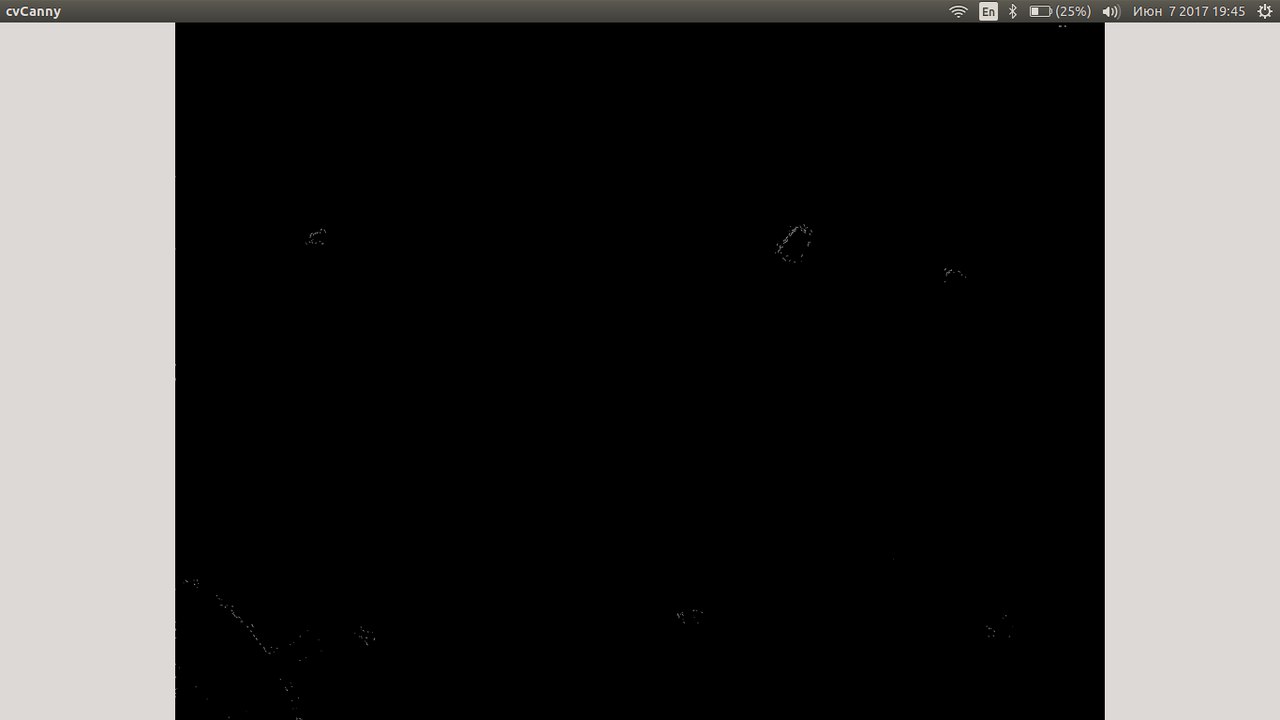
2 наступні параметри (threshold1 та treshold2) – пороги виділення границь. Якщо границі виділяються ледь помітно, різницю між цими дома параметрами треба збільшити. Проте чим більша різниця між ними, тим більші області виділяються як границі, що може призвести до хибного виділення об’єкту.

Параметри gridw і gridh – ширина і висота сітки. Ці параметри потрібні для визначення регіонів з деталями. Якщо одна деталь попаде відразу в декілька регіонів, то вона просто двічі розпізнається з близькими координатами. При переміщенні модулю, коли деталь попаде в той же регіон що і робочий носик, вона буде розпізнана як одна деталь з відповідними координатами. Якщо в один регіон попаде декілька деталей, то вони будуть неправильно розпізнані як одна. Тому цієї ситуації потрібно уникати. Тобто краще зробити сітку з меншими регіонами, але достатньо великими аби найбільша з деталей правильно розізналась в тому ж регіоні що і робочий носик. Зображення з накладеною сіткою на малюнку 9



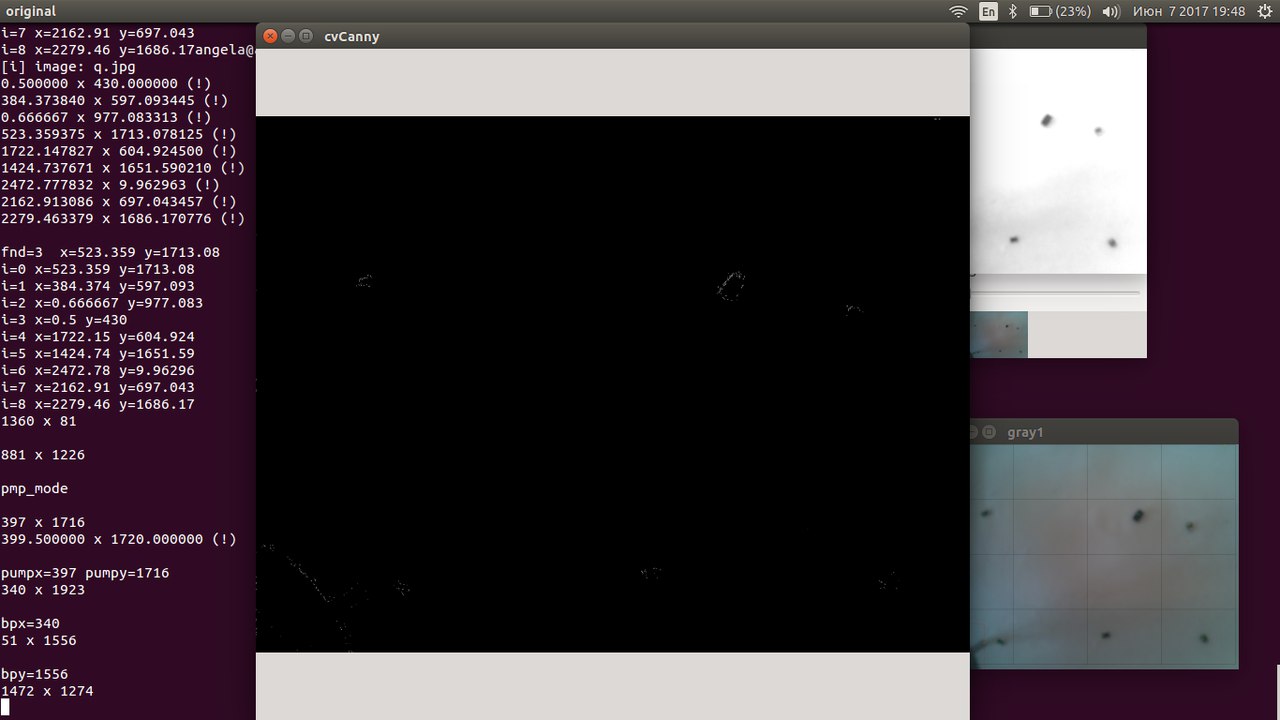
Мал. 9 – Зображення з накладеною сіткою

В результаті отримуємо зображення з границями (мал. 10)



Мал. 10 – Результуюче зображення

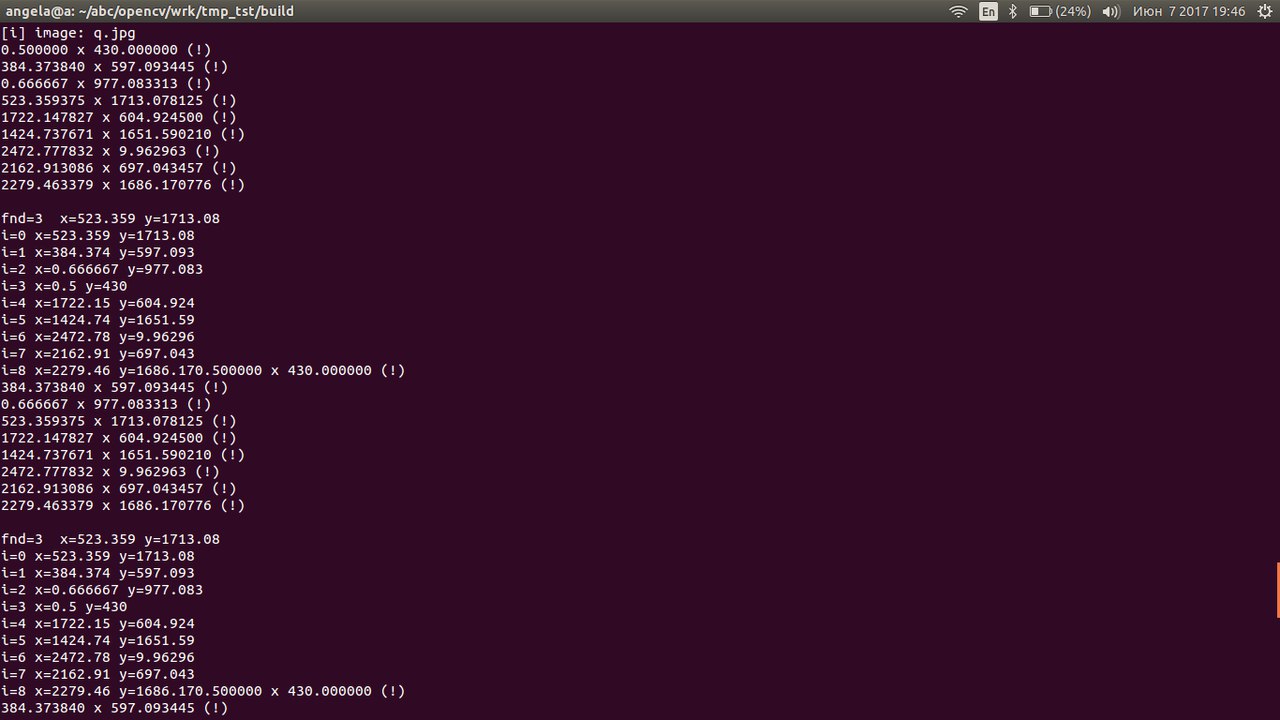
Для правильної обробки зображення необхідно з розглядуваних координат виключити координати з зображенням робочого носику. Це робиться наступним чином: на результуючому зображення кліком правої кнопки миші заходимо в режим pmp\_mode (мал. 11). Далі клікаємо на кінці самого робочого носику, нижній горизонтальній границі носику та його крайній вертикальній границі. Отримана квадратна область (позначено червоним на малюнку 9) виключається з розпізнавання. Наступним кліком лівої кнопки миші можна вийти з режиму pmp\_mode.



Мал. 11 – Виділення робочого носику

Також є можна визначити координату довільної точки на малюнку кліком лівої кнопки миші результуючому зображенні в відповідній точці (мал. 11).

В результаті на консоль виводяться координати всіх розпізнаних деталей, а також координати найближчої з них до робочого носику з рядком “fnd” (мал. 12).



Мал. 12 Отримання координат зображення

На мал. 9 жовтим виділений кінець робочого носику, червоним область що виключається з розпізнавання, зеленим – розпізнані деталі. Центр границі деталі визначається як середнє арифметичне точок границі.

### 2.3 Керування кроковими двигунами

В роботі використовувались як біполярні (для руху осей) так і уніполярний (для позиціонування деталі) крокові двигуни. Лістинг – додаток 3//write it

За керування біполярними кроковими двигунами відповідає клас my\_bp. З початку цим двигунам необхідно привести положення робочих площадок в точку O(0,0,0). Це робить метод void my\_bp::initb(). В цьому методі рух відбувається у напрямку механічних кінцевиків доки ці кінцевики не замкнуться. При замиканні вони подадуть відповідний сигнал на керуючий мікроконтролер, який по даному сигналу зупиняє рух відповідного крокового двигуна. Далі рух керується методом void my\_bp::mv(unsigned int num,bool to), де num – кількість кроків, to – напрям: якщо to==1, то це рух до відповідного механічного кінцевика, інакше – від нього.

За керування уніполярним кроковим двигуном відповідає клас my\_rot. Рух цього двигуна керується методом void my\_rot::mv(unsigned int num,bool to), де num – кількість кроків, to – напрям: якщо to==1, то це рух за годинниковою стрілкою, інакше – проти годинникової стрілки.

## 3. Подальші перспективи

В подальшому варто об’єднати програмну і апаратну частину модулю та протестувати їх спільну роботу.

Також існує ряд варіантів для покращення модулю:

1. Налагодити роботу UART між комп’ютером і керуючим мікроконтролером.
2. Покращення розпізнавання деталей: спробувати відмовитися від сітки і розпізнавати деталі по ступеню близькості сусідніх граничних точок.
3. Розпізнавати деталі з побудовою SIFT(Scale-invariant feature transform) дескрипторів.
4. Виконувати обробку зображення та розпізнавання деталей на Raspberry Pi і відповідно налагодити роботу UART між Raspberry Pi та керуючим мікроконтролером.
5. Замінити Raspberry Pi та Raspberry Pi Camera Module відповідно на STM32 та OV7670 і відповідно налагодити роботу UART між STM32 та комп’ютером і комп’ютером та керуючим мікроконтролером.
6. Замінити 2 джерела живлення на 1 з використанням подільника напруги і, можливо, стабілітрона.
7. Написати нейронну мережу для переміщення робочих площадок.
8. Написати модуль для виділення з креслень трасувань плат координати, позицію та тип елемента.

#!/bin/bash

while true

do

raspistill -o ./www/pic.jpg

echo "pic.jpg done"

sleep 2

done