МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**

Кафедра **ПЗ**

Лабораторна робота № 1

На тему: *"Дослідження роботи ультразвукового та інфрачервоного сенсорів відстані за допомогою середовища* **STM32CubeIDE** *та плати STM32F4 Discovery"*

З дисципліни: *"Основи інтернету речей"*

Львів – 2021

Тема роботи: *Дослідити особливості роботи ультразвукового та інфрачервоного сенсорів відстані за допомогою середовища* **STM32CubeIDE** *та плати STM32F4 Discovery*.

Мета роботи: Дослідити роботу сенсорів відстані пристроїв відображення інформації на прикладі символьного рідкокристалічного дисплею.

Теоретичні відомості

Ультразвуковий датчик відстані HC-SR04

Ультразвуковий далекомір HC-SR04 - це вміщені на одну плату приймач і передавач ультразвукового сигналу. Випромінювач генерує сигнал, який, відбившись від перешкоди, потрапляє на приймач. Вимірявши час, за який сигнал проходить до об'єкта і назад, можна оцінити відстань. Крім самих приймача і передавача, на платі знаходиться ще і необхідна обв'язка, щоб зробити роботу з цим датчиком простою і зручною.

Характеристики ультразвукового далекоміра HC-SR04:

- вимірюваний діапазон: від 2 до 500 см;
- точність: 0,3 см;
- кут огляду: менше 15 °;
- напруга живлення: 5 В.

Піни сенсора:

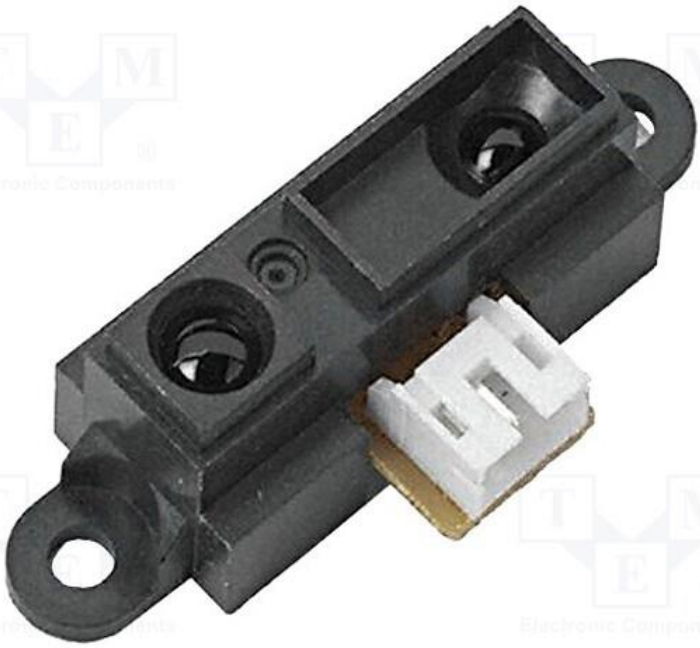
- VCC - живлення +5 В;
- Trig (T) – пін вхідного сигналу;
- Echo (R) - пін вихідного сигналу;
- GND - земля.

Послідовність дій для отримання даних така:

- подаємо імпульс тривалістю 10 мкс на пін Trig;
- всередині далекоміра вхідний імпульс перетворюється в 8 імпульсів частотою 40 кГц і надсилається вперед через випромінювач T;

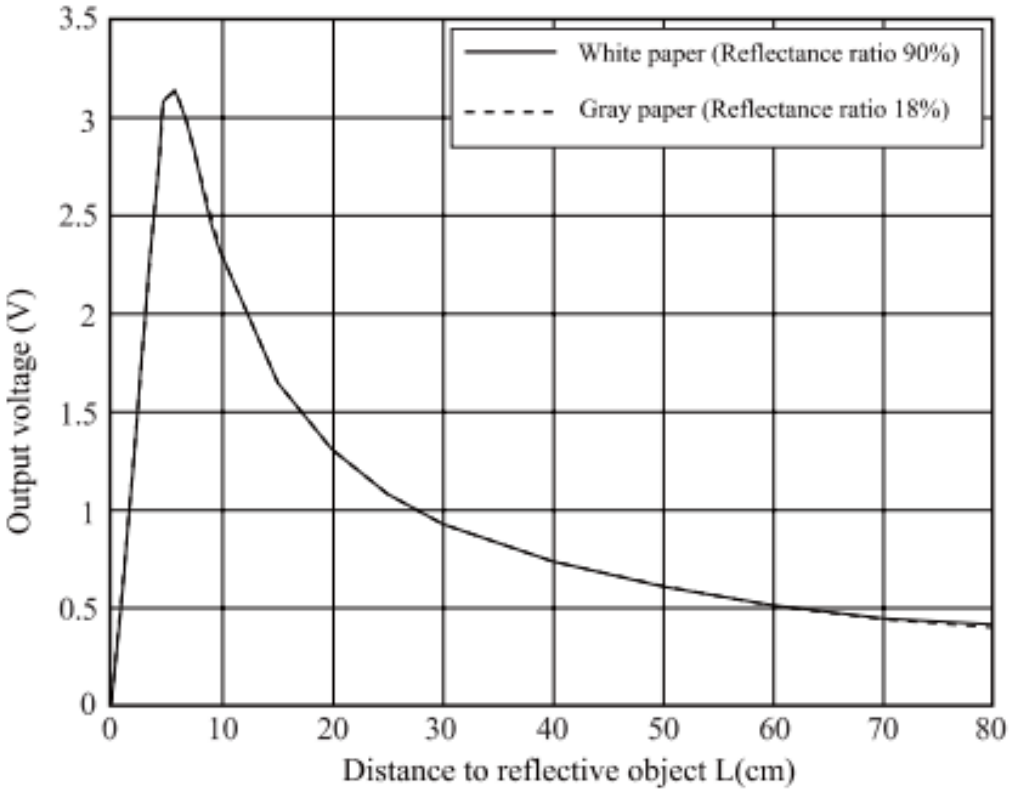
- дійшовши до перешкоди, послані імпульси відбиваються і приймаються приймачем R, в результаті отримуємо вихідний сигнал на виводі Echo;
- безпосередньо на стороні контролера переводимо отриманий сигнал в відстань за формулою: ширина імпульсу (мкс) / 58 = дистанція (см). Число 58 визначається із фізичного явища, що швидкість імпульс повинен прийти 2 відстані до об’єкта (туди і назад) зі швидкістю звуку. Це число може відрізнятися, якщо помістити сенсор в інше середовище.

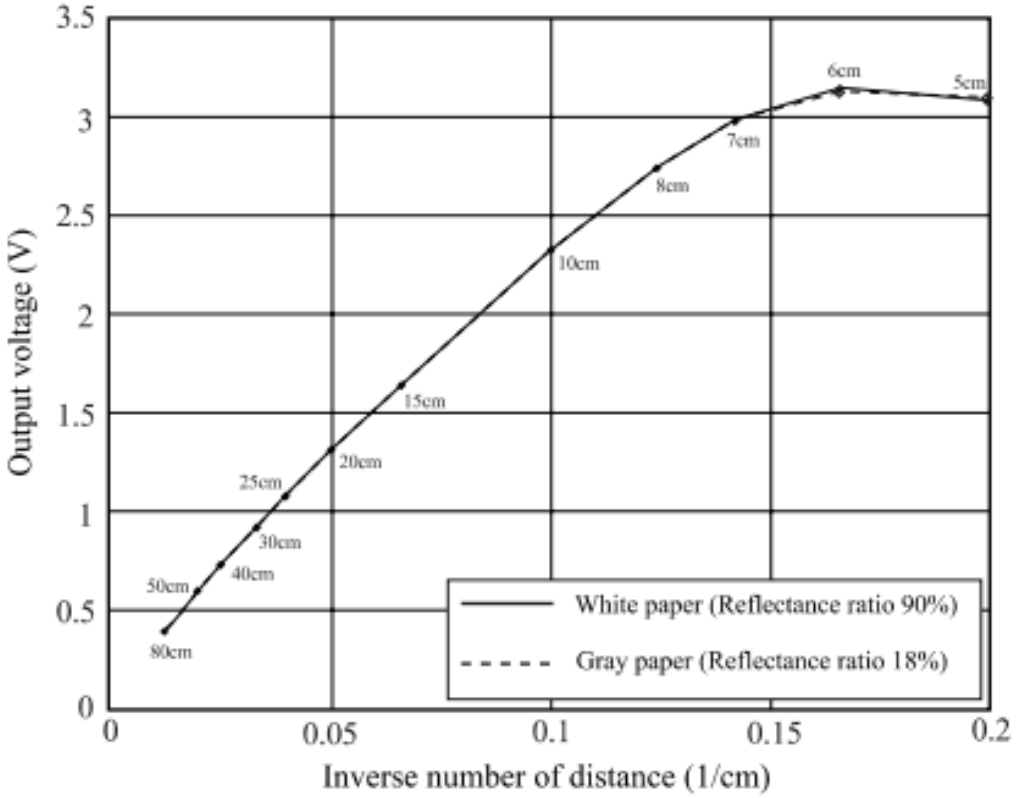
Аналоговий датчик відстані Sharp GP2Y0A21



Датчик GP2Y0A21 - це інфрачервоний датчик відстані від Sharp, який може виявити об'єкт, розташований на відстані від 10 до 80 см перед ним. Цей діапазон відстаней встановлений через лінійність залежності аналогового сигналу.

Датчик відстані GP2Y0A21 використовує інфрачервоне світло для обчислення відстані до об’єкта шляхом тріангуляції. Інфрачервоний світлодіод посилає світловий сигнал, невидимий неозброєним оком, який відбивається в присутності предмета. Стрічка з фоторезистом вловлює світло, що відбивається, і виводить кут відбиття, а отже, і відстань. Відповідно до документації датчика аналогове значення приходить в межах від 0 до 3,3 В.





Відповідно до верхнього графіка можна побачити, що графік відповідності аналогового сигналу до відстані не є монотонна, тому датчик можна використовувати не тільки для відстані від 0 до 6 см теж.

Характеристики інфрачервоного далекоміра:

- вимірюваний діапазон: від 10 до 80 см;
- напруга живлення: 5 В.

Піни сенсора:

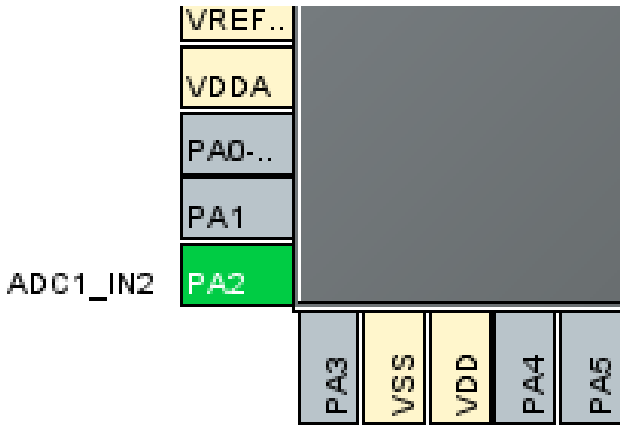
- VCC - живлення +5 В;
- Analog – пін вхідного сигналу;
- GND - земля.

Для роботи з аналоговими сенсорами обов’язкове використання фільтрів для запобігання стрибків значення.

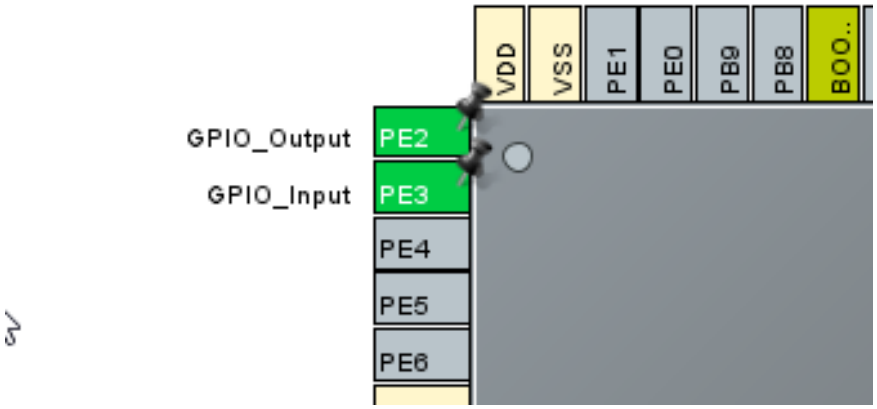
Створення проекту

Створимо проект, в якому буде задіяно 2 типи сенсорів і дисплей.

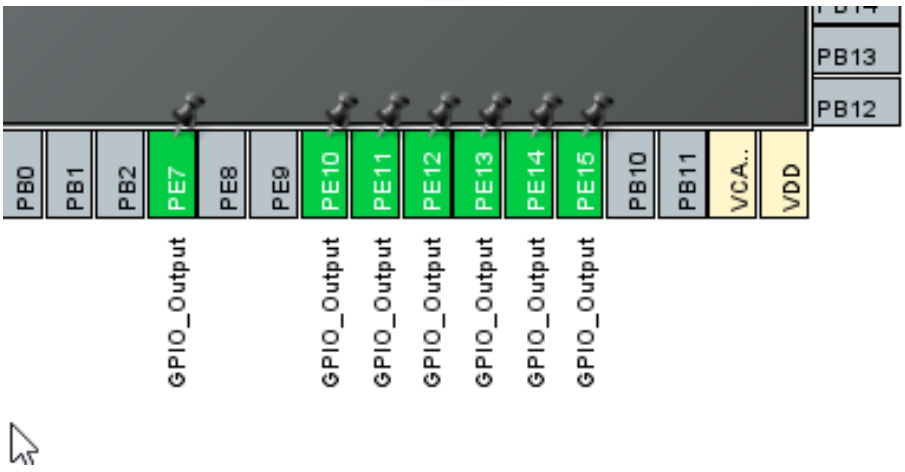
Для підключення інфрачервоного сенсору використаємо пін PA2, який відповідає за IN2 ADC1. Серед основних параметрів необхідно вказати розрядність результату ADC. В даному випадку використаємо 12 бітів.



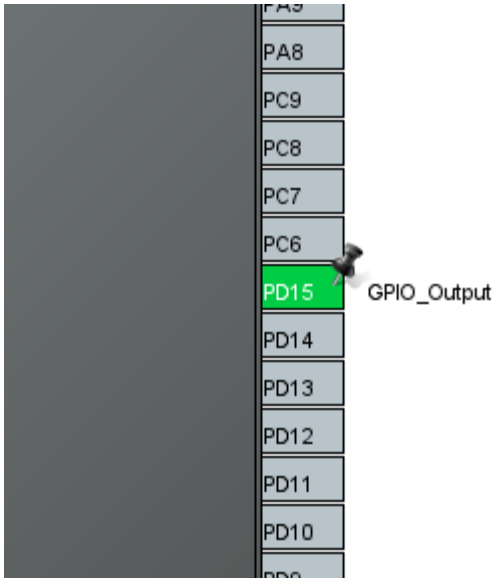
Для підключення акустичного датчика використаємо піни PE2 для Trig та PE3 для Echo. Необхідно налаштувати PE2 пін під вихід, а PE3 на вхід.



Також використаємо попередній проект з дисплеєм для виведення інформації.



Для визначення коректної роботи програми додамо блимаючий світлодіод, який буде показувати такт зчитування.



Робота з сенсорами потребує додаткових утиліт, які можуть використовуватися в будь-якому рівні програми. Наприклад це фільтр та ламана математична функція, які будуть використовуватися в даному проекті.

Код модуля ламаної математичної функції:

```
struct Math2DPointStruct
{
  int16_t x;
  int16_t y;
};

typedef struct Math2DPointStruct math_2d_point_t;

struct MathFunction2DObjStruct
{
  const math_2d_point_t* objP_data_table;
  uint16_t u16_table_size;
};

typedef struct MathFunction2DObjStruct math_function_2d_obj_t;

static int16_t math_function_get_fx(int16_t s16L_x, int16_t s16L_x1, int16_t s16L_y1,
  int16_t s16L_x2, int16_t s16L_y2);

void math_function_2d_construct(math_function_2d_obj_t* objPL_this,
  const math_2d_point_t* objPL_data, uint16_t u16L_dataSize)
{
  objPL_this->objP_data_table = objPL_data;
  objPL_this->u16_table_size  = u16L_dataSize;
}

int16_t math_function_2d_get_y(math_function_2d_obj_t* objPL_this, int16_t s16L_x)
{
  const math_2d_point_t* objPL_first_point = &objPL_this->objP_data_table[0];

  if(s16L_x < objPL_first_point->x)
  {
    return objPL_first_point->y;
  }

  const math_2d_point_t* objPL_last_point =
    &objPL_this->objP_data_table[objPL_this->u16_table_size - 1];

  if (s16L_x >= objPL_last_point->x)
  {
    return objPL_last_point->y;
  }

  const math_2d_point_t* objPL_cur_point  = objPL_first_point;
  const math_2d_point_t* objPL_next_point = objPL_first_point + 1;

  while (objPL_cur_point != objPL_last_point)
  {
    if (s16L_x < objPL_next_point->x)
    {
      return math_function_get_fx(s16L_x, objPL_cur_point->x,
        objPL_cur_point->y, objPL_next_point->x, objPL_next_point->y);
    }

    objPL_cur_point = objPL_next_point;
    objPL_next_point++;
  }
```

```
  // return 0 will never be executed
  return 0;
}

int16_t math_function_get_fx(int16_t s16L_x, int16_t s16L_x1, int16_t s16L_y1,
  int16_t s16L_x2, int16_t s16L_y2)
{
  return ((s16L_x - s16L_x1) * (s16L_y2 - s16L_y1)) / (s16L_x2 - s16L_x1) + s16L_y1;
}
```

Код фільтра:

```
typedef struct
{
  uint32_t u32_calc_value;
  uint8_t u8_filter_time;
  bool b_filter_inited;
} filter_t;

void filter_init(filter_t* objPL_this, uint8_t u8L_filter_time)
{
  objPL_this->u32_calc_value = 0;
  objPL_this->b_filter_inited = false;
  objPL_this->u8_filter_time = u8L_filter_time;
}

uint16_t filter_calc(filter_t* objPL_this, uint16_t u16L_value)
{
  if (!objPL_this->b_filter_inited)
  {
    objPL_this->u32_calc_value = u16L_value * objPL_this->u8_filter_time;
    objPL_this->b_filter_inited = true;
    return u16L_value;
  }

  objPL_this->u32_calc_value -= objPL_this->u32_calc_value / objPL_this-
>u8_filter_time;
  objPL_this->u32_calc_value += u16L_value;

  return (uint16_t)(objPL_this->u32_calc_value / objPL_this->u8_filter_time);
}
```

Для роботи з інфрачервоним сенсором створимо наступну структуру, яка використовуватиметься в функціях роботи з сенсором:

```
struct IrSensorStruct
{
  ADC_HandleTypeDef* objP_adc;
  math_function_2d_obj_t obj_func;
};
```

Для зчитування даних із інфрачервоного напишемо функції ініціалізації та зчитування.

```
#define MIN_ADC 460
#define MAX_ADC 3870

static const math_2d_point_t objPS_data[] =
{
  { MIN_ADC, 800 },
  { 776,     500 },
  { 931,     400 },
```

```
  { 1117,    300 },
  { 1365,    250 },
  { 1614,    200 },
  { 2017,    150 },
  { 2681,    100 },
  { 3413,    80  },
  { 3724,    70  },
  { MAX_ADC, 60  }
};

void ir_sensor_init(ir_sensor_t* objPL_this, ADC_HandleTypeDef* objPL_adc)
{
  objPL_this->objP_adc = objPL_adc;
  HAL_Delay(START_DELAY);

  math_function_2d_construct(&objPL_this->obj_func, objPS_data, 11);
}
```

Дана функція зберігає дані про ADC до внутрішніх даних об’єкта, а також ініціалізує дані для пошуку даних про температуру у відповідності до значення ADC.

Функція зчитування даних має наступний вигляд. Тут використано принцип блокування програми, тобто в час зчитування значення з датчика ніяка інша операція не може бути виконана.

Зазначимо, що значення ADC є нестійке і потребує фільтрування.

```
uint16_t ir_sensor_read(ir_sensor_t* objPL_this)
{
  filter_t objL_filter;
  filter_init(&objL_filter, FILTER_COUNT);

  uint16_t u16L_dist = 0;
  for (uint8_t u8L_i = 0; u8L_i < FILTER_COUNT; u8L_i++)
  {
    HAL_ADC_Start(objPL_this->objP_adc);
    HAL_ADC_PollForConversion(objPL_this->objP_adc, 0xFFFF);
    u16L_dist = filter_calc(&objL_filter, HAL_ADC_GetValue(objPL_this->objP_adc));
  }

  return ir_sensor_get_distance(objPL_this, u16L_dist);
}

uint16_t ir_sensor_get_distance(ir_sensor_t* objPL_this, uint16_t u16L_adc_val)
{
    if (u16L_adc_val >= MAX_ADC || u16L_adc_val <= MIN_ADC)
  {
    return TOO_FAR_DIST;
  }

  return math_function_2d_get_y(&objPL_this->obj_func, (int16_t)u16L_adc_val);
}
```

Для зчитування даних із акустичного сенсора напишемо функції ініціалізації та зчитування. Також створимо структуру яка зберігатиме внутрішні параметри об’єктів для роботи з функціями.

```
struct AcousticSensorStruct
{
  GPIO_TypeDef * objP_trig_port;
  uint16_t u16_trig_pin;
  GPIO_TypeDef * objP_echo_port;
  uint16_t u16_echo_pin;
};
```

Функція зберігає у внутрішній стан об’єкта піни, що використовуються для роботи з датчиком.

```
void acoustic_sensor_init(acoustic_sensor_t* objPL_this, GPIO_TypeDef *
objPL_trig_port,
  uint16_t u16L_trig_pin, GPIO_TypeDef * objPL_echo_port, uint16_t u16L_echo_pin)
{
  objPL_this->objP_echo_port = objPL_echo_port;
  objPL_this->objP_trig_port = objPL_trig_port;
  objPL_this->u16_echo_pin = u16L_echo_pin;
  objPL_this->u16_trig_pin = u16L_trig_pin;
}

Функція зчитування:
uint16_t acoustic_sensor_read(acoustic_sensor_t* objPL_this)
{
  uint32_t u32L_timer = delay_get_us_tick();

  HAL_GPIO_WritePin(objPL_this->objP_trig_port, objPL_this->u16_trig_pin,
GPIO_PIN_SET);
  while (delay_get_us_tick() - u32L_timer < TRIG_TIME);

  HAL_GPIO_WritePin(objPL_this->objP_trig_port, objPL_this->u16_trig_pin,
GPIO_PIN_RESET);
  while (HAL_GPIO_ReadPin(objPL_this->objP_echo_port, objPL_this->u16_echo_pin) ==
GPIO_PIN_RESET);

  u32L_timer = delay_get_us_tick();
  while (HAL_GPIO_ReadPin(objPL_this->objP_echo_port, objPL_this->u16_echo_pin) ==
GPIO_PIN_SET);

  uint32_t u32L_duration = delay_get_us_tick() - u32L_timer;

  return acoustic_sensor_convert_duration(u32L_duration);
}

uint16_t acoustic_sensor_convert_duration(uint32_t u32L_duration)
{
  return (u32L_duration * 10) / 58;
}
```

Для зчитування необхідно підняти тригер пін на 10 мікросекунд після чого опустити пін та очікувати підняття рівня напруги на піні exo. Після отримання високого сигналу записуємо час підняття та очікуємо опускання піна. Як тільки пін опуститься вираховуємо час скільки часу був піднятий пін та конветуємо це значення у відстань.

Застосуємо написані функції в ході програми. Перед вычним циклом використаємо наступну ініціалізацію:

```
ir_sensor_t obj_ir_sensor;
acoustic_sensor_t obj_ac_sensor;
delay_init(&htim6);
initLCD();
ir_sensor_init(&obj_ir_sensor, &hadc1);
```

А також в цикл додамо зчитування даних та виведення їх на екран.

```
clear();
char cPL_str[16];
uint16_t u16L_dist = ir_sensor_read(&obj_ir_sensor);
sprintf(cPL_str, "%d", u16L_dist);
setCursor(0, 0);
lcdString(cPL_str);
u16L_dist = acoustic_sensor_read(&obj_ac_sensor);
setCursor(0, 1);
memset(cPL_str, 0, 16);
sprintf(cPL_str, "%d", u16L_dist);
lcdString(cPL_str);
HAL_Delay(1000);
HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
```

ХІД РОБОТИ

1. Ознайомитись з технічними особливостями застосування ультразвукового та інфрачервоного сенсорів тиску
2. Створити проект для одночасного визначення відстані обома сенсорами з виводом значень на екран дисплею.
3. Провести вимірювання відстані для предмету чорного кольору, білого кольору та предмету з м’якою структурою (поролон).
4. Проаналізувати отримані результати та пояснити розбіжності.