Зертханалық жұмыс №1
Электромобиль микроконтроллерінің кірістері мен шығыстарын бағдарламалау

Пән: Электромобильдерге арналған микроконтроллерлік жүйелерді бағдарламалау

1. Жұмыстың мақсаты

Электромобильдің қарапайым электрондық жүйесі мысалында микроконтроллердің цифрлық және аналогтық кіріс сигналдарымен және атқарушы құрылғылармен өзара әрекеттесуінің негізгі принциптерін зерттеу.

Зертханалық жұмыс барысында студент:

- STM32 микроконтроллеріне арналған жобаны құруды және іске қосуды;
- GPIO цифрлық кірістері мен шығыстарын баптауды;
- цифрлық кіріс күйін оқуды;
- ADC көмегімен аналогтық сигналды оқуды;
- ADC мәнін физикалық шамаға түрлендіруді;
- акселератор педалінің күй сигналын модельдеуді;
- температура сигналын модельдеуді;
- қарапайым басқару логикасын жүзеге асыруды;
- диагностикалық ақпаратты шығаруды;
- келесі тізбектің жұмыс логикасын түсінуді үйренуі тиіс:

датчик → ADC / GPIO → микроконтроллер → бағдарлама → шешім → басқару сишналы → атқарушы құрылғы.

2. Қолданылатын орта

Зертханалық жұмысты орындау үшін тек:

- Wokwi онлайн симуляторы;
- STM32F103C8T6 Blue Pill виртуалды тақтасы;
- Wokwi ішіндегі бағдарлама редакторы;
- Serial Monitor

қолданылады.
Қосымша бағдарламаларды орнату қажет емес.

3. Маңызды ескерту

STM32 Blue Pill кейбір жағдайларда Wokwi-дің қарапайым Add Part мәзірінде көрінбеуі мүмкін. Сондықтан студенттер жұмысты оқытушы берген дайын STM32F103C8 Wokwi жобасынан бастайды.
https://wokwi.com/projects/476622661513474049

Студенттің міндеті:

1. дайын жобаны ашу;
2. жобаның көшірмесін жасау;
3. қажетті электрондық компоненттерді қосу;
4. бағдарламалық кодты өзгерту;
5. симуляцияны іске қосу;
6. нәтижелерді тексеру.

4. Қолданылатын компоненттер

Зертханалық жұмыста:

- STM32F103C8T6;
- 2 потенциометр;
- 1 батырма немесе switch;
- READY LED;
- FAULT LED;
- резисторлар

қолданылады.

5. Қысқаша теориялық бөлім

5.1. Микроконтроллер электромобиль жүйесінің бөлігі

Микроконтроллер датчиктерден ақпарат алады, оны бағдарламаға сәйкес өңдейді және басқарушы сигналдарды қалыптастырады.

Жалпы құрылымы:



6. GPIO

GPIO — General Purpose Input/Output — жалпы мақсаттағы цифрлық кірістер мен шығыстар. GPIO екі негізгі режимде жұмыс істейді.

INPUT

Микроконтроллер цифрлық сигналды қабылдайды:

```
Button / Switch
      ↓
     GPIO
      ↓
     MCU
```

Сигнал екі логикалық күйдің бірін қабылдай алады:
0 → LOW
1 → HIGH

OUTPUT

Микроконтроллер басқарушы сигналды қалыптастырады:

```
MCU
 ↓
GPIO
 ↓
LED / Relay / Driver
```

Нақты электромобильде цифрлық сигналдар келесі мақсаттарда пайдаланылуы мүмкін:

- қосқыш күйін анықтау/состояния переключателя;
- жүйені іске қосуға рұқсат беру / разрешения запуска;
- контактор күйін бақылау;
- релелерді басқару;
- enable-сигналдарын басқару;
- ақау сигналдарын беру / передачи сигналов неисправности.

7. ADC

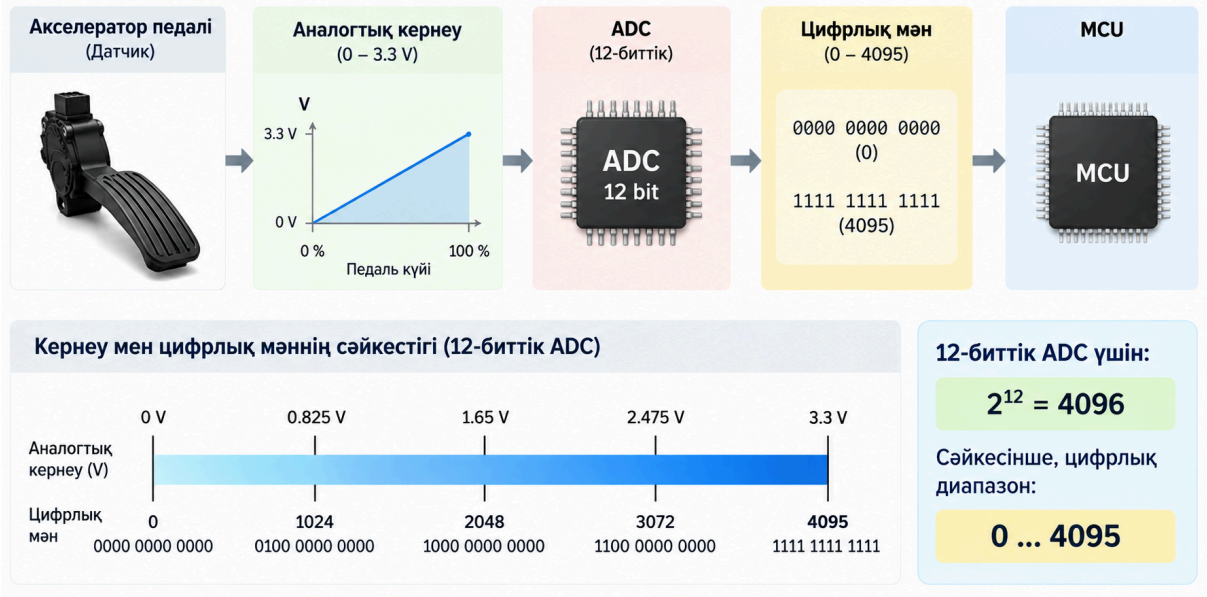
Автомобильдегі көптеген физикалық шамалар аналогтық болып табылады:

- температура;
- кернеу;
- ток;
- қысым;
- педаль күйі.

Ал микроконтроллер цифрлық мәндермен жұмыс істейді. Сондықтан:

ADC — Analog-to-Digital Converter — аналогтық-цифрлық түрлендіргіш қолданылады.

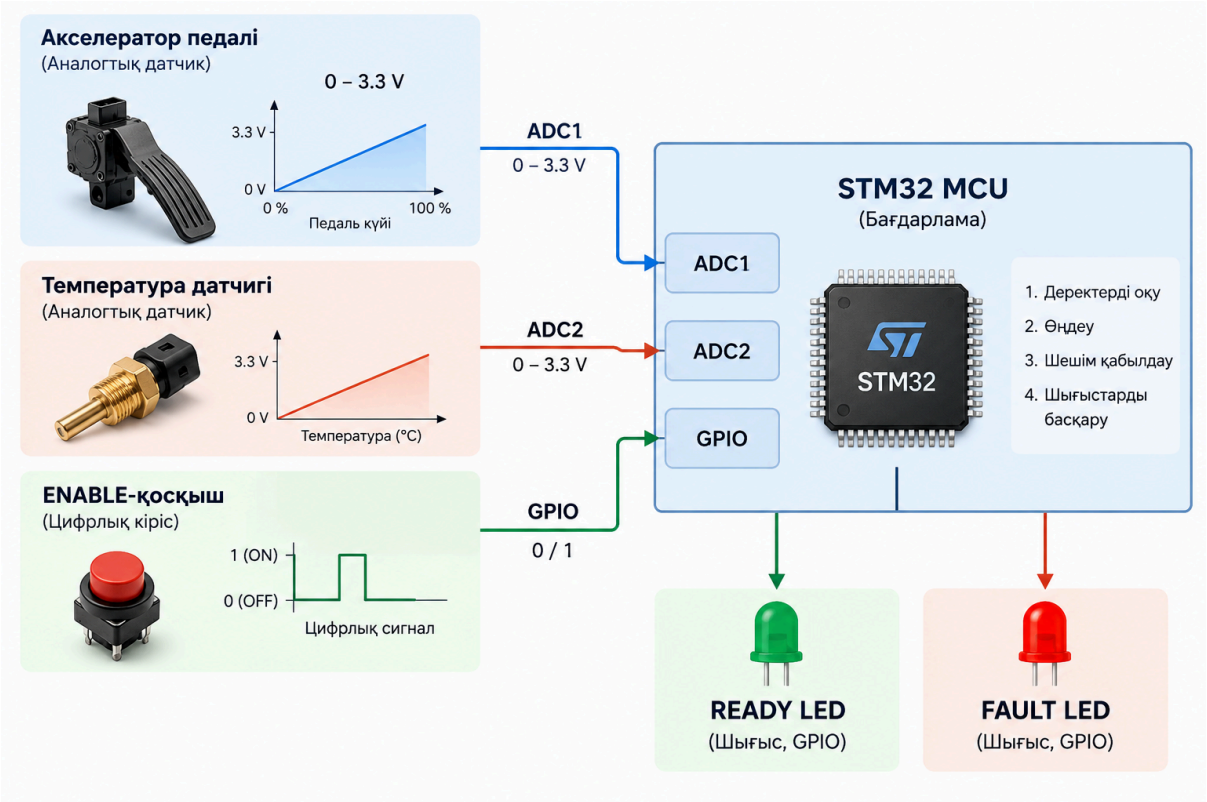




8. Зерттеу объектісі

Бұл зертханалық жұмыста Vehicle Control Unit — VCU кіріс модулінің қарапайым үлгісі модельденеді. Келесі элементтер пайдаланылады:

1. акселератор педалінің күй датчигі;
2. температура датчигі;
3. ENABLE цифрлық ауыстырып-қосқышы;
4. READY индикаторы;
5. FAULT индикаторы.



8.1. Датчиктерді модельдеу

Аналогтық датчиктерді модельдеу үшін потенциометрлер қолданылады.

Потенциометр №1 – Accelerator Pedal Position Sensor акселератор педалінің күй датчигін модельдейді

Потенциометр №2 — Battery Temperature Sensor температура датчигін модельдейді

Оқу мақсатында келесі диапазонды қабылдаймыз:
ADC тің ең төмен мәні → температура датчигінің 0°C на сәйкес
ADC тің ең жоғары мәні → температура датчигінің 100°C мәніне сәйкес
Бұл — шартты оқу шкаласы, нақты автомобильдік температура датчигінің сипаттамасы емес / Это условная учебная шкала, а не характеристика реального автомобильного температурного датчика.

9. 1-тапсырма. Wokwi жобасын іске қосу

Оқытушы берген STM32F103C8 жобасын ашыңыз. Жобаның жеке көшірмесін жасаңыз.

Жоба атауын өзгертіңіз: Lab_01_EV_IO және өзіңіздің атыңызды қосып сақтаңыз

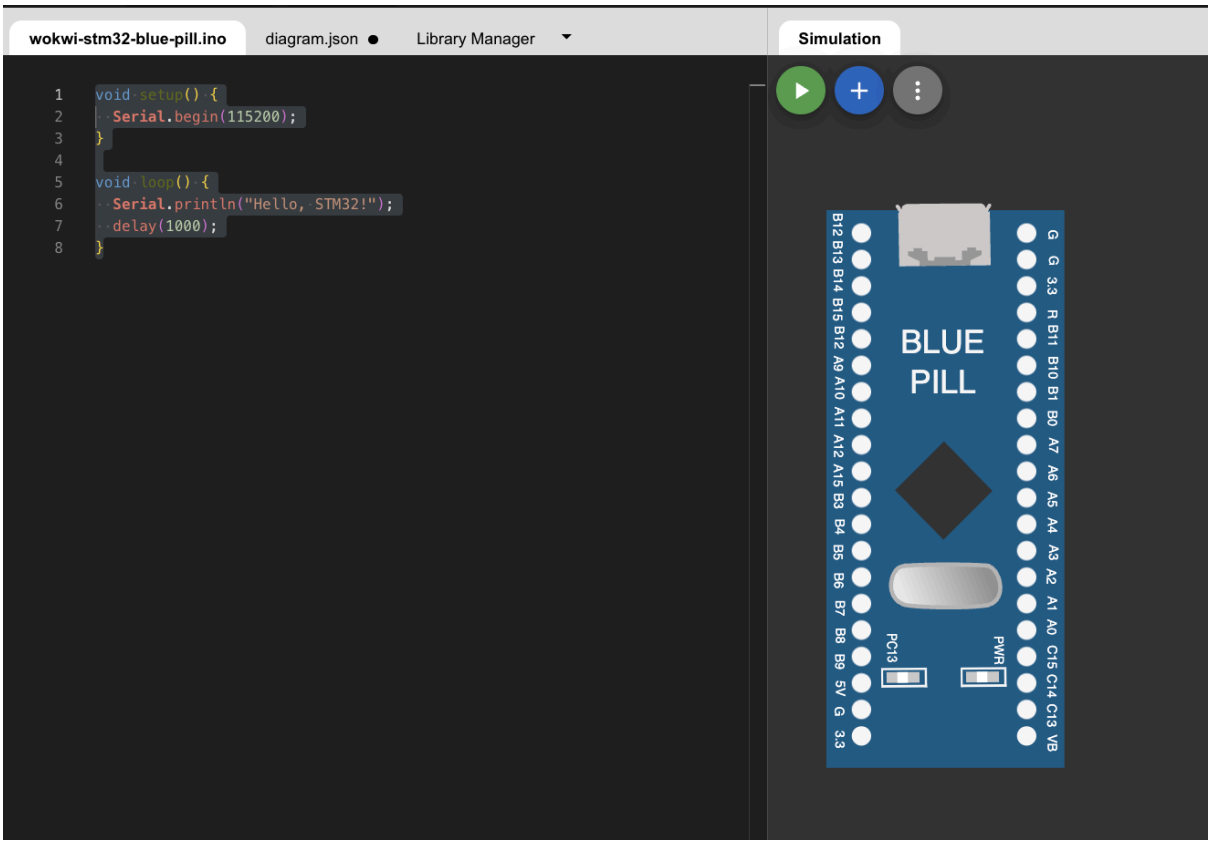
STM32 тақтасының симуляциясын іске қосыңыз.

Микроконтроллердің бағдарламасы қатесіз іске қосылғанына көз жеткізіңіз.

Жасыл ▶ Start Simulation батырмасын басыңыз. Бағдарлама қателерсіз компиляцияланып, симуляция іске қосылғанына көз жеткізіңіз. Serial Monitor терезесінде Hello, STM32! хабарламасының пайда болуын тексеріңіз.

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  Serial.println("Hello, STM32!");
  delay(1000);
}
```



10. 2-тапсырма. Цифрлық шығысты зерттеу
READY LED-ті микроконтроллердің цифрлық шығысына қосыңыз.

Мысалы:

```
#define READY_LED PC13

void setup()
{
   pinMode(READY_LED, OUTPUT);
}

void loop()
{
   digitalWrite(READY_LED, HIGH);
   delay(500);

   digitalWrite(READY_LED, LOW);
   delay(500);
}
```

Симуляцияны іске қосыңыз.

Тексеру

Келесі мәндерді қолданыңыз:

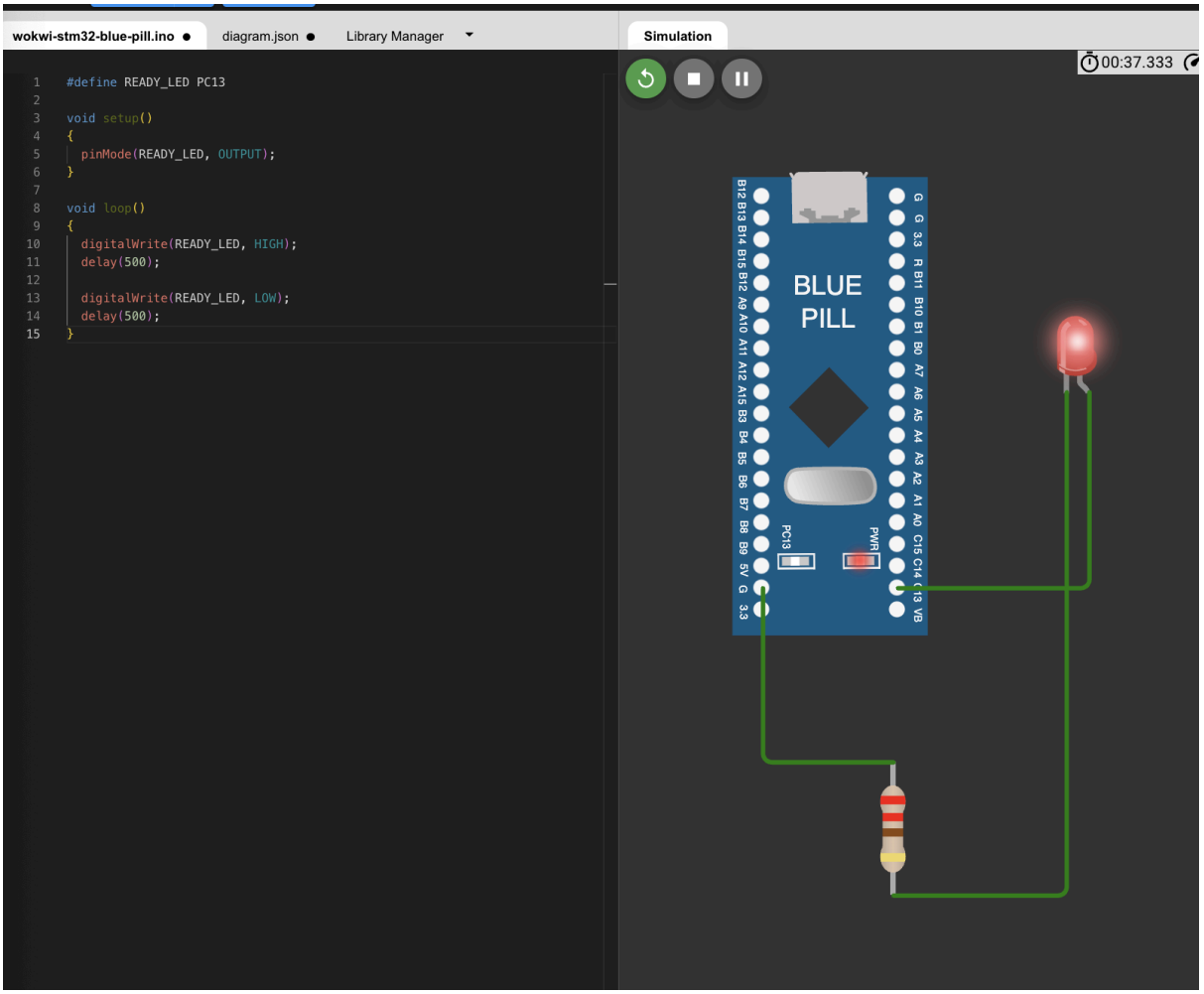
delay(100)

delay(500)

delay(1000)

delay(2000)

LED жұмысының өзгеруін бақылаңыз.



11. 3-тапсырма. Цифрлық кіріс
ENABLE батырмасын немесе switch қосыңыз.

```
#define ENABLE_PIN PB12
#define READY_LED  PC13

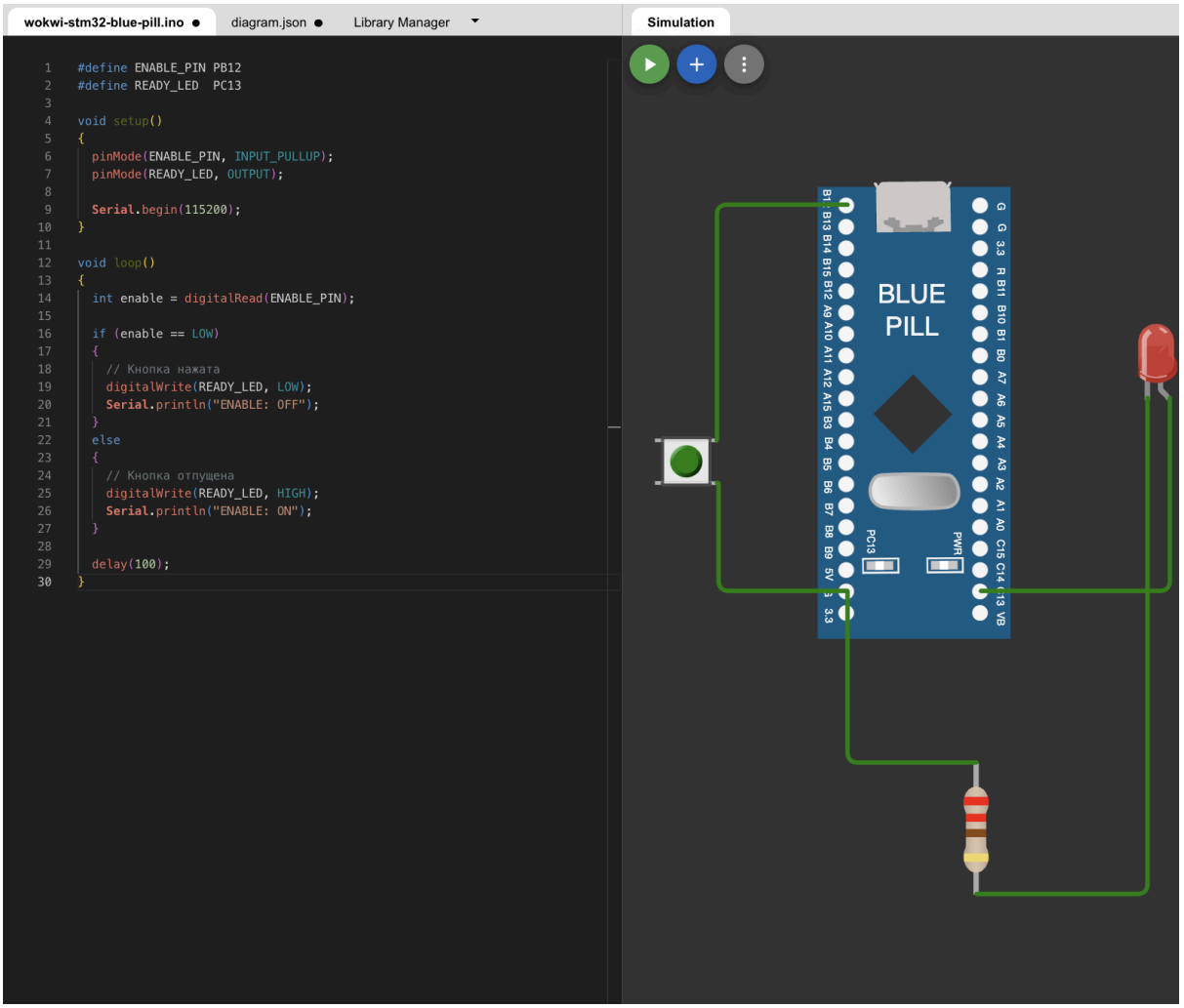
void setup()
{
  pinMode(ENABLE_PIN, INPUT_PULLUP);
  pinMode(READY_LED, OUTPUT);

  Serial.begin(115200);
}

void loop()
{
  int enable = digitalRead(ENABLE_PIN);
```

```
  if (enable == LOW)
  {
    // Кнопка нажата
    digitalWrite(READY_LED, LOW);
    Serial.println("ENABLE: OFF");
  }
  else
  {
    // Кнопка отпущена
    digitalWrite(READY_LED, HIGH);
    Serial.println("ENABLE: ON");
  }

  delay(100);
}
```



Мына логикамен не болады?

```
#define ENABLE_PIN PB12
#define READY_LED  PC13

void setup()
{
  pinMode(ENABLE_PIN, INPUT_PULLUP);
  pinMode(READY_LED, OUTPUT);

  Serial.begin(115200);
}

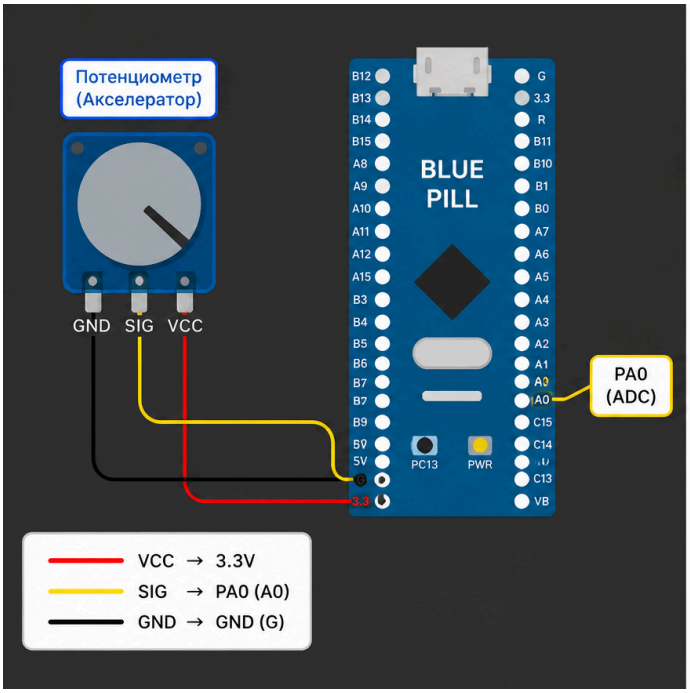
void loop()
{
  int enable = digitalRead(ENABLE_PIN);

  if (enable == LOW)
  {
    // Кнопка ENABLE нажата
    digitalWrite(READY_LED, HIGH);
    Serial.println("ENABLE: ON | READY: ON");
  }
  else
  {
    // Кнопка ENABLE отпущена
    digitalWrite(READY_LED, LOW);
    Serial.println("ENABLE: OFF | READY: OFF");
  }

  delay(100);
}
```

13. 4-тапсырма. Акселератор педалін модельдеу

Бірінші потенциометрді Accelerator Pedal Position Sensor ретінде пайдаланыңыз. Оны Микроконтроллер ішінде орналасқан ADC кірісіне қосыңыз. Мысалы: PA0




```
#define PEDAL_PIN PA0

void setup()
{
  Serial.begin(115200);

  analogReadResolution(12);
}

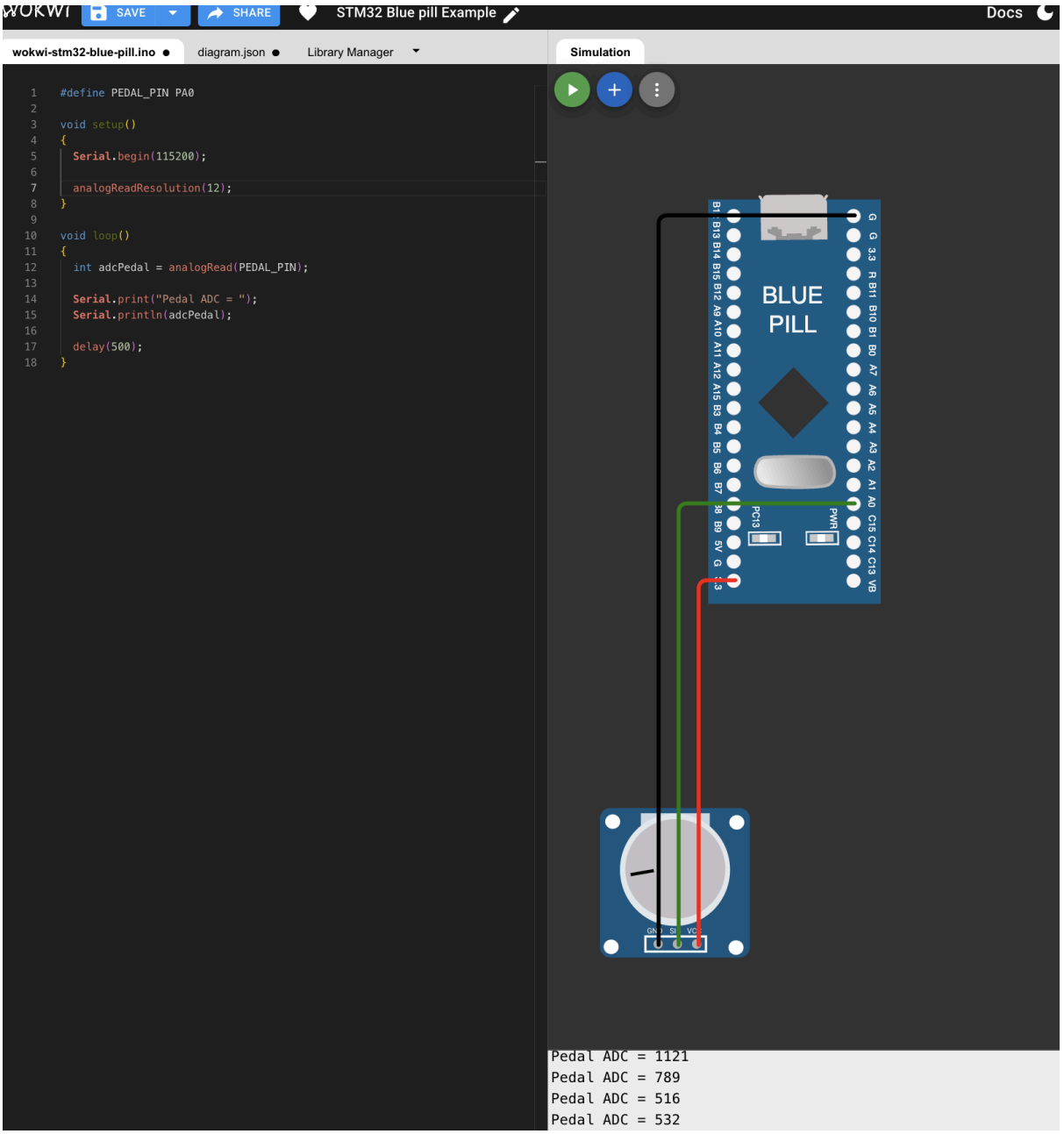
void loop()
{
  int adcPedal = analogRead(PEDAL_PIN);

  Serial.print("Pedal ADC = ");
  Serial.println(adcPedal);

  delay(500);
}
```

14. ADC мәнін оқу

Бағдарламада: int adcPedal = analogRead(PA0); мәнін алыңыз. Serial Monitor терезесіне шығарыңыз: Serial.println(adcPedal); Потенциометрді өзгертіңіз. ADC мәнінің қалай өзгеретінін бақылаңыз.



Зертханалық жұмыста потенциометр акселератор педалінің орналасу датчигін имитациялайды. Студент потенциометрді бұрған кезде, бұл «жүргізуші газ педалін басып немесе босатып жатыр» дегенді білдіреді. Бірақ потенциометр ADC қызметін атқармайды.

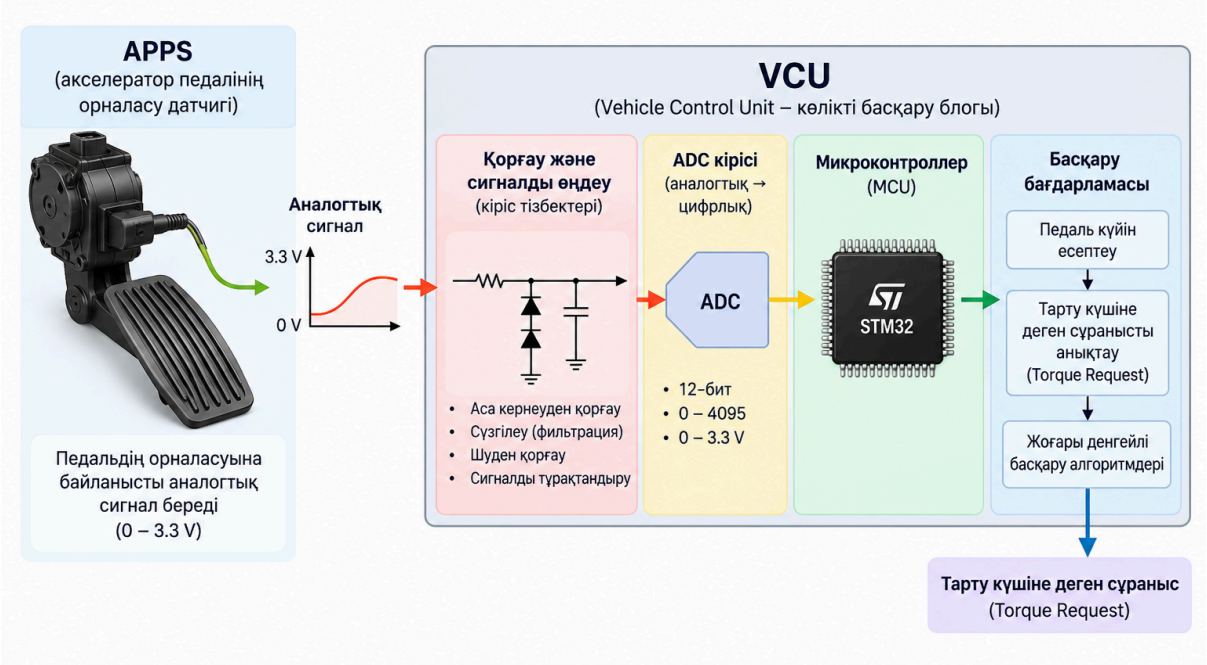
!Потенциометр бұл жерде датчик қызметін атқарып тұр. Студент потенциометрді бұрады → педальді имитациялайды → датчикте кернеу өзгереді → STM32-нің кіріктірілген ADC-і (Аналогты сандық түрлендіргіш) кернеуді цифрлық мәнге түрлендіреді → бағдарлама педальдің орналасу күйін анықтайды.

Нақты электромобильде (EV) кәдімгі потенциометрдің орнына Accelerator Pedal Position Sensor (APPS) — акселератор педалінің орналасу датчигі қолданылады.

Қауіпсіздікті қамтамасыз ету үшін әдетте педальдің орналасуын анықтайтын екі тәуелсіз немесе өзара салыстырылатын арна пайдаланылады. Контроллер екі арнадан да сигналдарды қабылдап, олардың бір-біріне сәйкестігін тексереді. Тек осы тексеруден кейін ғана жүргізушінің тарту күшіне деген сұранысын басқару үшін пайдаланады.

Нақты электромобиль жүйелерінде де ADC қолданылады. Егер акселератор педалінің датчигі аналогтық сигнал берсе, контроллер бағдарламасы бұл сигналмен жұмыс істей алуы үшін оны алдымен цифрлық мәнге түрлендіру қажет.

Бұл ретте ADC әдетте VCU жанында орналасқан жеке құрылғы болмайды. Ол автомобильдік микроконтроллердің перифериялық модулі болуы мүмкін — дәл сіздер қолданатын STM32 ішіндегі ADC сияқты. Сонымен қатар, ADC кірісіне дейін сигналды қорғауға және өңдеуге арналған кіріс тізбектері қолданылады.



15. Педаль күйін пайызбен есептеу

ADC мәнін: 0–100 % диапазонына түрлендіру қажет.

Жалпы формула:

$$Pedal\% = \frac{ADC}{ADC_{max}} \times 100$$

$$ADC_{MAX} = 2^{12} - 1 = \boxed{4095}$$

```
#define PEDAL_PIN PA0

void setup()
{
  Serial.begin(115200);

  // ADC ажыратымдылығы – 12 бит
  analogReadResolution(12);
}

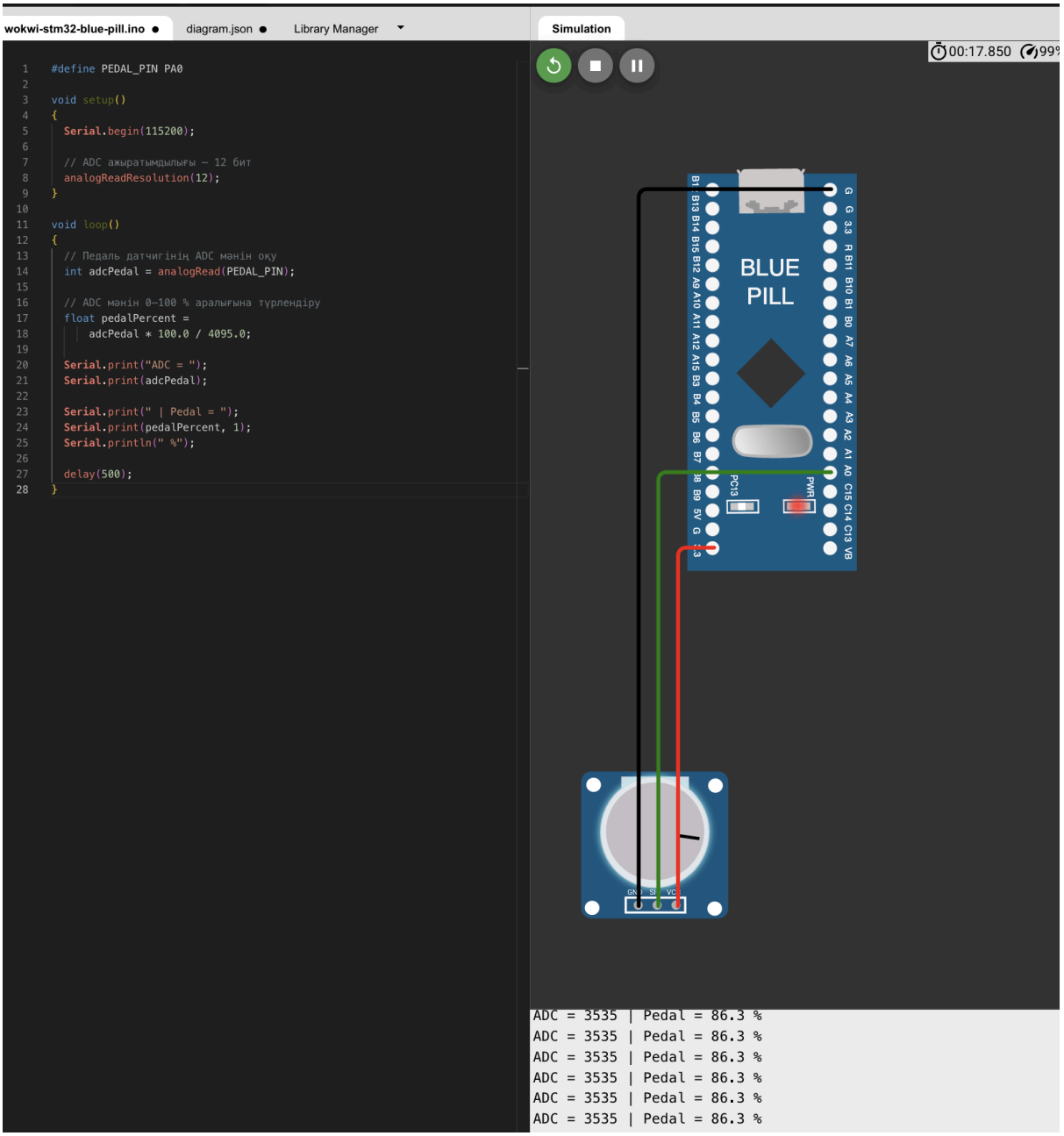
void loop()
{
  // Педаль датчигінің ADC мәнін оқу
  int adcPedal = analogRead(PEDAL_PIN);

  // ADC мәнін 0–100 % аралығына түрлендіру
  float pedalPercent =
      adcPedal * 100.0 / 4095.0;

  Serial.print("ADC = ");
  Serial.print(adcPedal);

  Serial.print(" | Pedal = ");
  Serial.print(pedalPercent, 1);
  Serial.println(" %");

  delay(500);
}
```



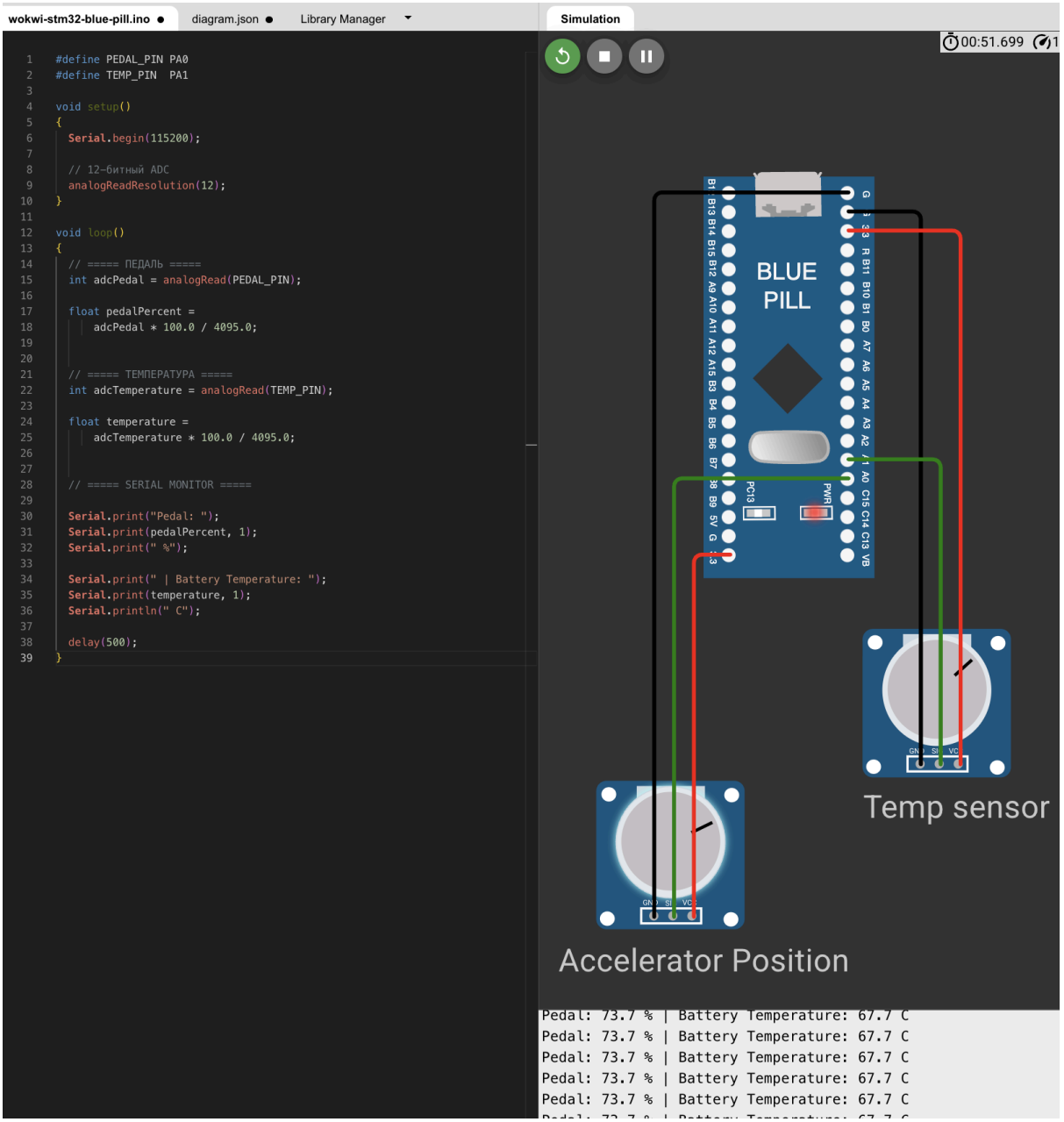
16. Педаль күйін тексеру

Потенциометрді бірнеше күйге орнатыңыз. Кестені толтырыңыз

Күй	ADC	Педаль, %
Minimum		≈ 0
1/8		
1/4		≈ 25
1/2		≈ 50

3/4		≈ 75
Maximum		≈ 100

Нәтижелерді есепке енгізіңіз.



17. 5-тапсырма. Батарея температурасын модельдеу

Екінші потенциометрді температура датчигі ретінде пайдаланыңыз. Мысалы: Battery Temperature Sensor

Оқу моделінде: Minimum → 0°C Maximum → 100°C деп қабылданады. Бұл нақты батарея температура датчигінің сипаттамасы емес, тек оқу үшін алынған шартты шкала.

```
#define PEDAL_PIN PA0
#define TEMP_PIN  PA1

void setup()
{
  Serial.begin(115200);

  // 12-битный ADC
  analogReadResolution(12);
}

void loop()
{
  // ===== ПЕДАЛЬ =====
  int adcPedal = analogRead(PEDAL_PIN);

  float pedalPercent =
      adcPedal * 100.0 / 4095.0;


  // ===== ТЕМПЕРАТУРА =====
  int adcTemperature = analogRead(TEMP_PIN);

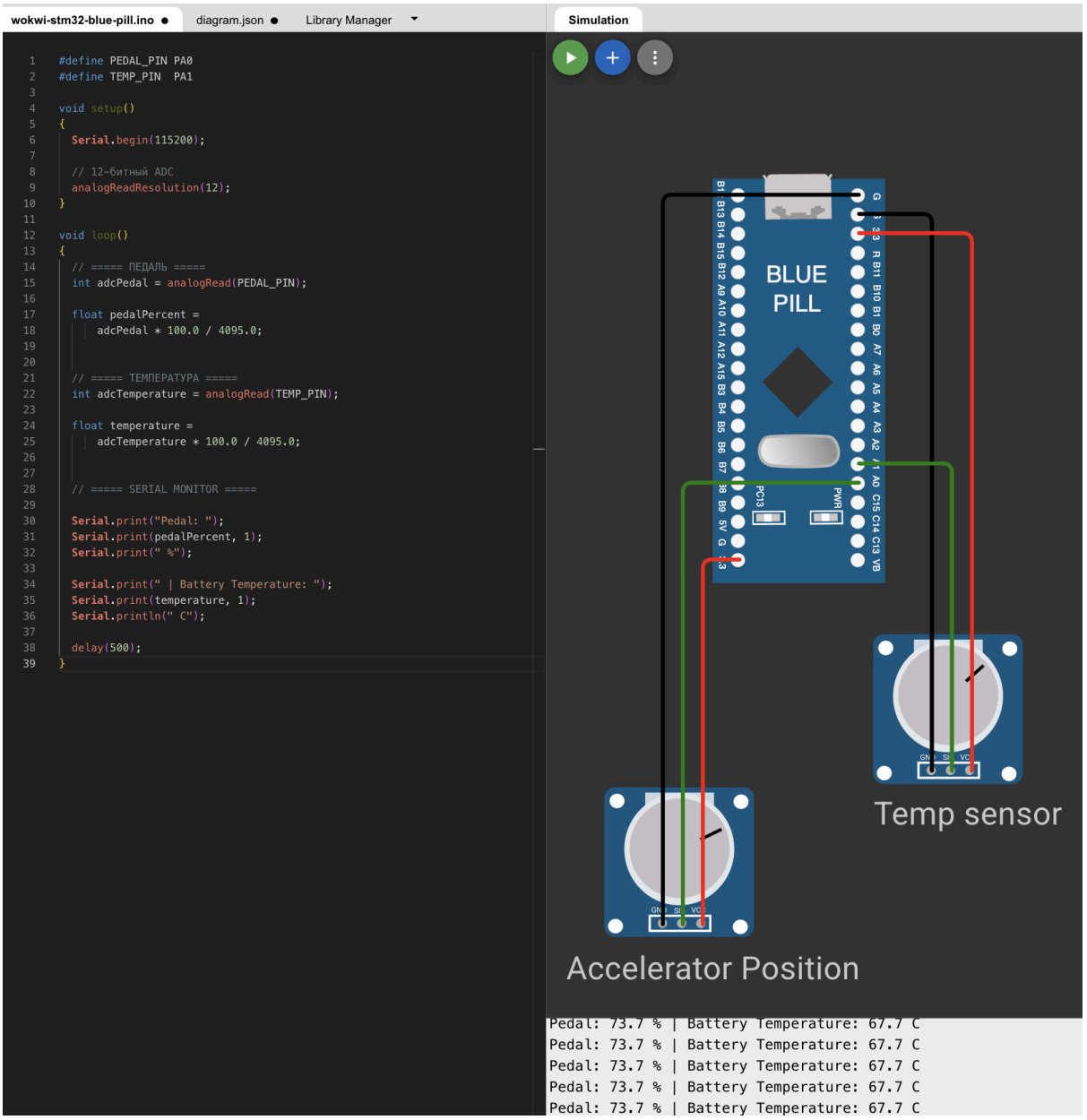
  float temperature =
      adcTemperature * 100.0 / 4095.0;


  // ===== SERIAL MONITOR =====

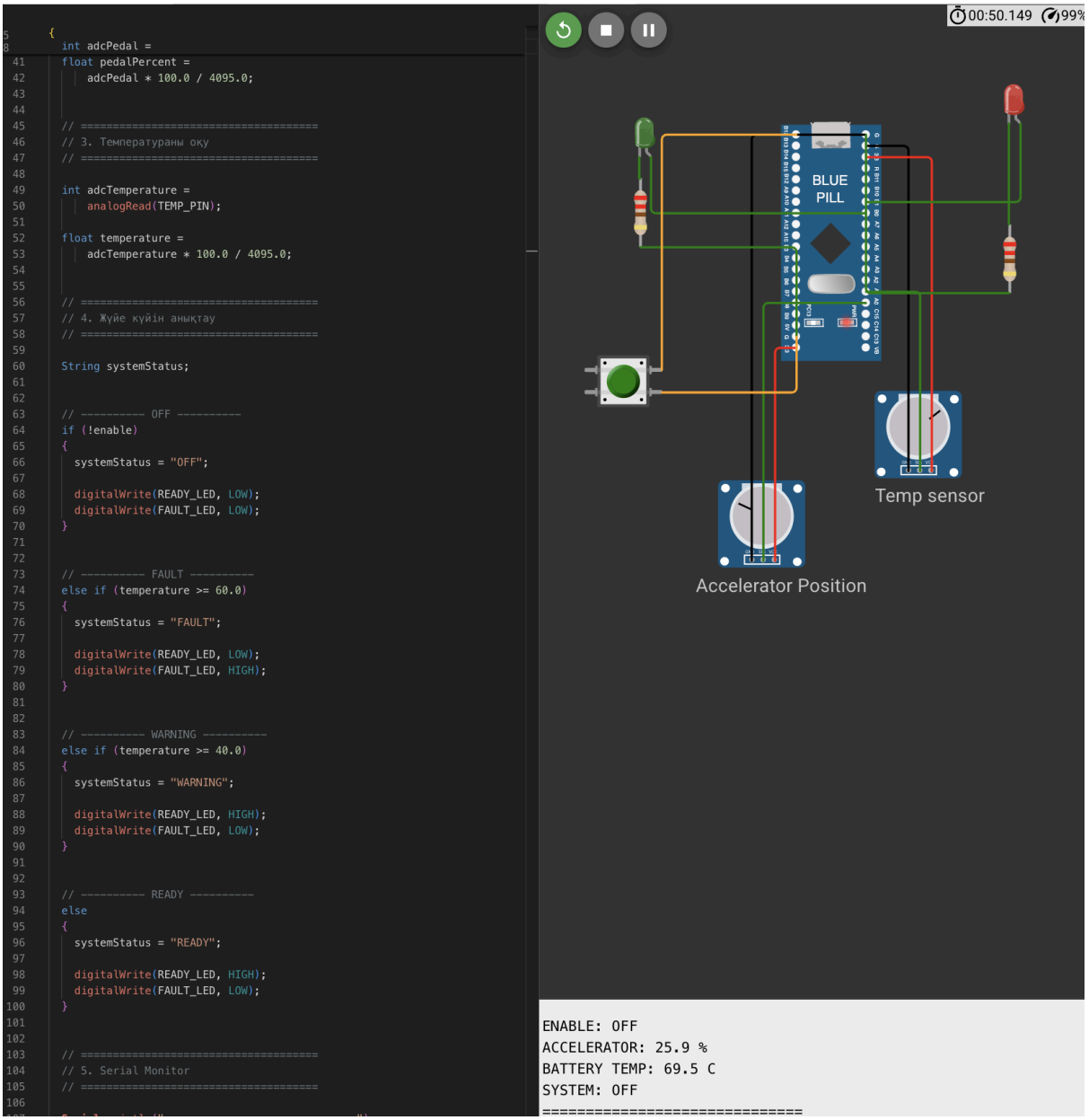
  Serial.print("Pedal: ");
  Serial.print(pedalPercent, 1);
  Serial.print(" %");

  Serial.print(" | Battery Temperature: ");
  Serial.print(temperature, 1);
  Serial.println(" C");

  delay(500);
}
```



Ақпаратты шамамен әр: 3000 ms сайын жаңартыңыз. Нені көрдіңіз?



18. 6-тапсырма. Жүйе күйлерін енгізу

Төрт күй пайдаланылады:

OFF
READY
WARNING
FAULT

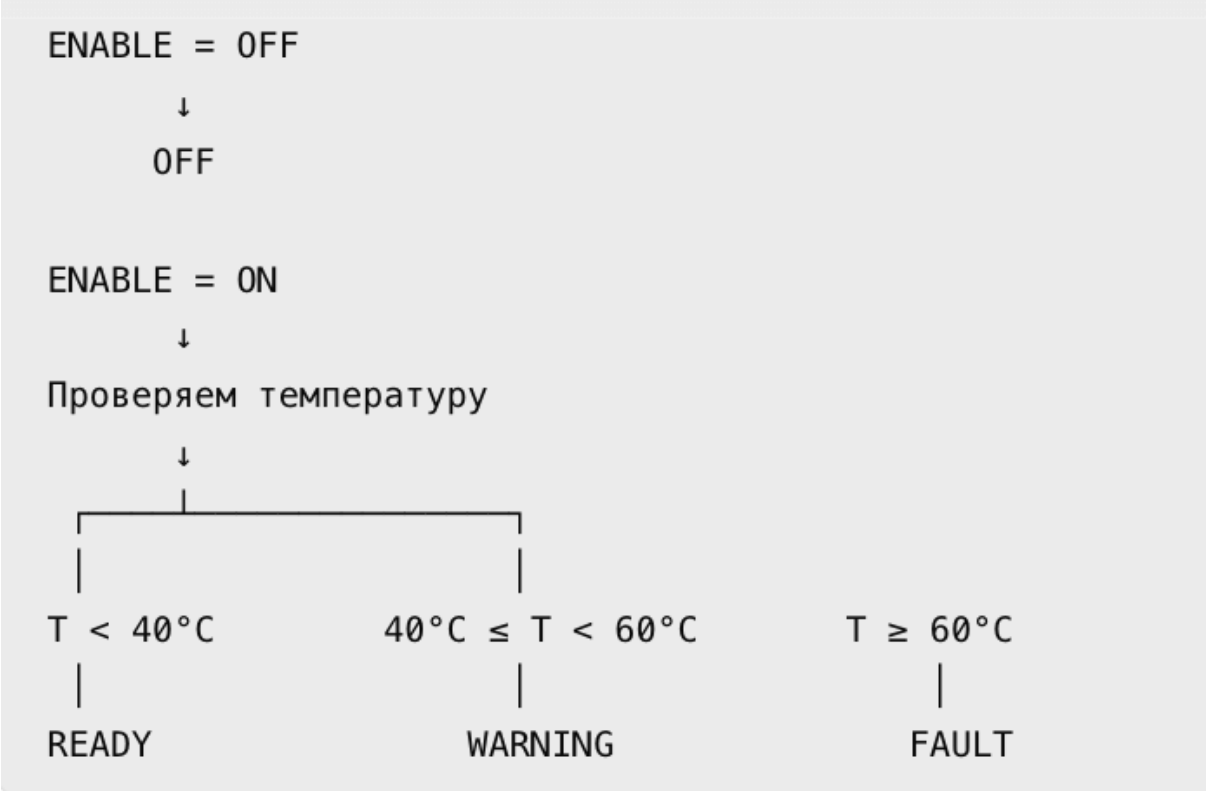
<u>OFF күйі</u>

Егер: ENABLE = OFF
болса: SYSTEM = OFF , READY LED = OFF , FAULT LED = OFF

READY күйі
Егер: ENABLE = ON және Temperature < 40°C
болса: SYSTEM = READY , READY LED = ON , FAULT LED = OFF

WARNING күйі
Егер: $40^{\circ}C \leq T < 60^{\circ}C$
болса: SYSTEM = WARNING

FAULT күйі
Егер: $T \geq 60^{\circ}C$
болса: SYSTEM = FAULT , READY LED = OFF , FAULT LED = ON



#define ENABLE_PIN PB12

#define PEDAL_PIN PA0
#define TEMP_PIN PA1

#define READY_LED PB0

```
#define FAULT_LED PB1


void setup()
{
  Serial.begin(115200);

  pinMode(ENABLE_PIN, INPUT_PULLUP);

  pinMode(READY_LED, OUTPUT);
  pinMode(FAULT_LED, OUTPUT);

  // STM32F103 ADC = 12 bit
  analogReadResolution(12);
}


void loop()
{
  // ========================================
  // 1. ENABLE оқу
  // ========================================

  bool enable =
     (digitalRead(ENABLE_PIN) == LOW);


  // ========================================
  // 2. Акселераторды оқу
  // ========================================

  int adcPedal =
     analogRead(PEDAL_PIN);

  float pedalPercent =
     adcPedal * 100.0 / 4095.0;


  // ========================================
  // 3. Температураны оқу
  // ========================================

  int adcTemperature =
```

```
    analogRead(TEMP_PIN);

float temperature =
    adcTemperature * 100.0 / 4095.0;


// ========================================
// 4. Жүйе күйін анықтау
// ========================================

String systemStatus;


// ---------- OFF ----------
if (!enable)
{
  systemStatus = "OFF";

  digitalWrite(READY_LED, LOW);
  digitalWrite(FAULT_LED, LOW);
}


// ---------- FAULT ----------
else if (temperature >= 60.0)
{
  systemStatus = "FAULT";

  digitalWrite(READY_LED, LOW);
  digitalWrite(FAULT_LED, HIGH);
}


// ---------- WARNING ----------
else if (temperature >= 40.0)
{
  systemStatus = "WARNING";

  digitalWrite(READY_LED, HIGH);
  digitalWrite(FAULT_LED, LOW);
}
```

```
  // ---------- READY ----------
  else
  {
    systemStatus = "READY";

    digitalWrite(READY_LED, HIGH);
    digitalWrite(FAULT_LED, LOW);
  }


  // ======================================
  // 5. Serial Monitor
  // ======================================

  Serial.println("================================");
  Serial.println("EV CONTROLLER STATUS");
  Serial.println();

  Serial.print("ENABLE: ");
  Serial.println(enable ? "ON" : "OFF");

  Serial.print("ACCELERATOR: ");
  Serial.print(pedalPercent, 1);
  Serial.println(" %");

  Serial.print("BATTERY TEMP: ");
  Serial.print(temperature, 1);
  Serial.println(" C");

  Serial.print("SYSTEM: ");
  Serial.println(systemStatus);

  Serial.println("================================");
  Serial.println();


  delay(500);
}
```

Енді студент барлық төрт күйді қолмен тексере алады.

ENABLE	Температура	SYSTEM	READY LED	FAULT LED
OFF	кез келген	OFF	OFF	OFF
ON	< 40°C	READY	ON	OFF
ON	40–<60°C	WARNING	ON	OFF
ON	≥ 60°C	FAULT	OFF	ON

Мысалы, ENABLE батырмасын басып, екінші потенциометрді шамамен 30% деңгейіне орнатамыз:
ENABLE: ON
ACCELERATOR: 45.2 %
BATTERY TEMP: 30.1 C
SYSTEM: READY
Бұл жағдайда READY LED жанады.

Температураны модельдейтін потенциометрді 40%-дан жоғары бұрамыз:
ENABLE: ON
ACCELERATOR: 45.2 %
BATTERY TEMP: 48.3 C
SYSTEM: WARNING

Жүйе WARNING күйіне өтеді, бірақ жұмысын жалғастырады.
Потенциометрді 60%-дан жоғары бұрамыз:
ENABLE: ON
ACCELERATOR: 45.2 %
BATTERY TEMP: 67.4 C
SYSTEM: FAULT
Бұл жағдайда READY LED сөнеді, ал FAULT LED жанады.

Жеке тапсырмалар

Әр студент өз нұсқасына сәйкес екінші потенциометрді белгілі бір датчиктің аналогтық сигналы ретінде модельдейді. ADC мәнін берілген физикалық шамаға түрлендіріп, жүйенің READY, WARNING, FAULT күйлерін анықтаңыз.

№	Модельденетін датчик	Өлшеу диапазоны	READY	WARNING	FAULT
1	Батарея температурасы	0–100 °C	< 40°C	40–60°C	≥ 60°C
2	Электр қозғалтқышының температурасы	0–150 °C	< 80°C	80–110°C	≥ 110°C
3	Инвертор температурасы	0–120 °C	< 70°C	70–90°C	≥ 90°C
4	Батарея кернеуі	250–450 V	320–420 V	280–320 V немесе 420–440 V	< 280 V немесе > 440 V
5	Батарея заряды — SOC	0–100 %	> 30%	10–30%	< 10%
6	Салқындату сұйықтығының температурасы	0–120 °C	< 70°C	70–90°C	≥ 90°C
7	Электр қозғалтқышының айналу жиілігі	0–12000 rpm	< 8000 rpm	8000–10000 rpm	≥ 10000 rpm
8	DC-link кернеуі	0–500 V	300–420 V	250–300 V немесе 420–450 V	< 250 V немесе > 450 V
9	Батарея тогы	0–300 A	< 150 A	150–220 A	≥ 220 A
10	Салқындату жүйесінің қысымы	0–5 bar	1.5–3.5 bar	1.0–1.5 bar немесе 3.5–4.0 bar	< 1.0 bar немесе > 4.0 bar

Бақылау сұрақтары

1. GPIO дегеніміз не?
2. GPIO Input және Output арасындағы айырмашылық қандай?
3. ADC не үшін қажет?
4. Аналогтық және цифрлық сигналдардың айырмашылығы қандай?
5. ADC разрядтылығы дегеніміз не?
6. 12-биттік ADC неше мәнді ажырата алады?
7. ADC мәнін физикалық шамаға қалай түрлендіруге болады?
8. Потенциометр датчикті қалай модельдейді?
9. Акселератор педалінің сигналы не үшін қажет?

10. ENABLE сигналының қызметі қандай?
11. READY күйі дегеніміз не?
12. WARNING күйі дегеніміз не?
13. FAULT күйі дегеніміз не?
14. INPUT_PULLUP режимінің ерекшелігі қандай?
15. MCU мен ECU арасындағы айырмашылық қандай?
16. VCU қандай қызмет атқарады?
17. Электромобильде қандай аналогтық датчиктер қолданылуы мүмкін?