|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | Изображение выглядит как зарисовка, эмблема, символ, герб  Содержимое, созданное искусственным интеллектом, может быть неверным. |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА** – **Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт искусственного интеллекта (ИИ) | |
| Кафедра промышленной информатики | |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1** | |
| **по дисциплине «Программирование киберфизических систем»** | |
|  | |
| Выполнил студент группы ЭФБО-01-22 | Столяров Даниил Шикхарович |
| Проверил доцент кафедры ПИ ИИИ | Благовещенский Владислав Германович |

**Задание**

Необходимо реализовать модель пешеходного перехода с двумя связанными светофорами: один светофор отвечает за регулирование пешеходов, второй светофор отвечает за регулирование проезда транспорта.

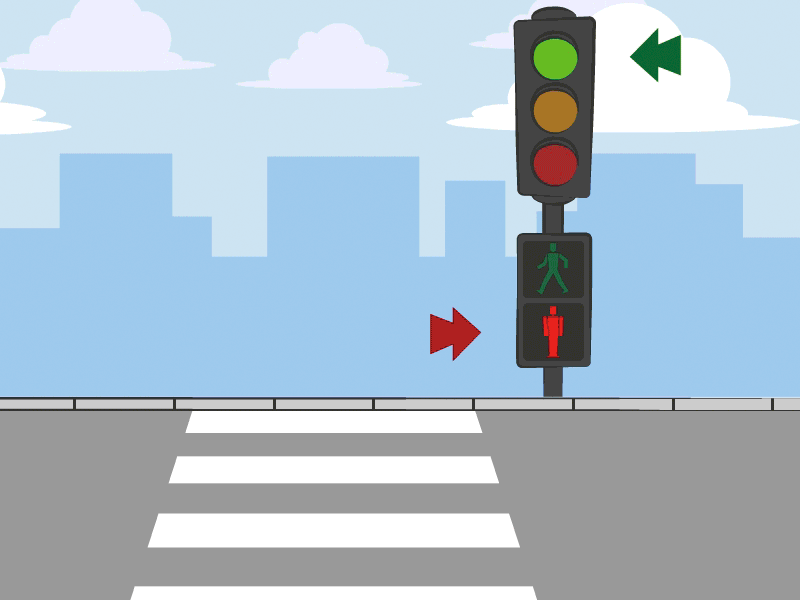


Рис. 1 – изображение пешеходного перехода

В базовом варианте реализуем светофор для транспорта с помощью трёх LED красного, жёлтого и синего цветов, а для пешеходов – LCD дисплей с двумя пресетами (красный и зеленый человечек). Для зелёного человечка добавим анимацию бега (2 картинки которые меняются).

Также добавим возможность регулировать время, в течение которого для пешеходов активен зелёный цвет с помощью отдельного регулятора; кнопку вызова для пешеходов, которая будет сообщать системе о желании человека перейти через дорогу.

**Установка и настройка**

Будем использовать локальную симуляцию от Wokwi в VScode. Для этого установим плагин Wokwi Simulator.

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 2 – Расширение Wokwi Simulator

Выполнять сборку тоже будем локально (основной плюс – не надо стоять в очереди, как при компиляции через браузер)

Потребуется установить Arduino-CLI.

Скачиваем из https://docs.arduino.cc/arduino-cli/installation/ версию для Windows-x64, распаковываем в любое удобное место, не забываем добавить в **PATH**.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3 – arduino-cli установлен

В отличие от браузерного wokwi, в Visual Studio Code можно установить расширение для IntelliSense (которое подтянет все библиотеки из Arduino)

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 4 – расширение для IntelliSense

В настройках нужно указать путь к Arduino-CLI.

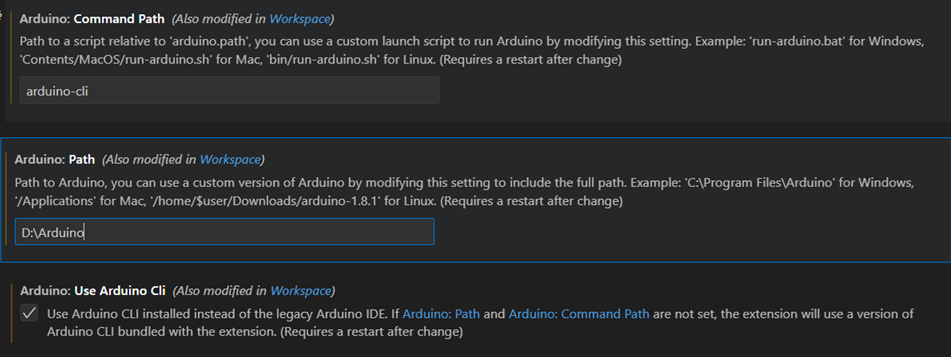


Рис. 5 – настройки Arduino расширения

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 6 – доступные команды расширения

Потребуется также запустить Arduino: Initialize, Arduino: Board Config, Arduino: Rebuild IntelliSense Configuration.

Выбираем в конфигурации платы необходимую плату для практической работы.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 7 – конфигурация платы

Бонус: это расширение также умеет загружать прошивку на плату, при необходимости можно будет без проблем запускать код на настоящей плате, а не в симуляции.

Далее необходимо сконфигурировать Arduino-CLI , чтобы он загрузил пакеты совместимости с STM32.

Листинг 1 – команды для arduino-cli

arduino-cli config init

arduino-cli config set network.connection\_timeout 180s

arduino-cli config add board\_manager.additional\_urls https://github.com/stm32duino/BoardManagerFiles/raw/main/package\_stmicroelectronics\_index.json

arduino-cli core update-index

arduino-cli core install STMicroelectronics:stm32

Теперь, если всё правильно настроено, при нажатии alt+ctrl+I запустить build проекта. После успешного build появятся файлы. elf и.hex в папке build/.

Wokwi сейчас о них ничего не знает, поэтому нужно создать в корне проекта файл wokwi.toml и поместить туда содержимое

Листинг 2 – конфигурация wokwi

[wokwi]

version = 1

firmware = "build/TrafficLight.ino.hex"

elf = "build/TrafficLight.ino.elf"

В бесплатной версии Wokwi нельзя редактировать симуляцию графически (только сам .json).

Обойдём это ограничение, добавив все нужные для симуляции компоненты непосредственно в браузерную симуляцию. Далее можно просто скопировать .json из браузерного Wokwi в корень проекта и продолжать работу локально.

Листинг 3 – файл конфигурации платы по умолчанию

{

"version": 1,

"author": "Puri Puri",

"editor": "wokwi",

"parts": [

{ "type": "board-st-nucleo-c031c6", "id": "nucleo", "top": 10.43, "left": -0.98, "attrs": {} }

],

"connections": [ [ "$serialMonitor:TX", "nucleo:PA3", "", [] ], [ "$serialMonitor:RX", "nucleo:PA2", "", [] ] ],

"dependencies": {}

}

Готово, теперь можно запускать симуляцию через диаграмму.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 8 – запуск симуляции в VS Code

**Сборка схемы**

Собираем схему согласно заданию. В симуляции VCC и GND пины дисплея можно не подключать.

Листинг 4 – diagram.json

{

"version": 1,

"author": "Puri Puri",

"editor": "wokwi",

"parts": [

{ "type": "board-st-nucleo-c031c6", "id": "nucleo", "top": 10.43, "left": -0.98, "attrs": {} },

{

"type": "wokwi-led",

"id": "led1",

"top": -1.2,

"left": 342.6,

"rotate": 90,

"attrs": { "color": "red" }

},

{

"type": "wokwi-led",

"id": "led2",

"top": 56.4,

"left": 342.6,

"rotate": 90,

"attrs": { "color": "yellow" }

},

{

"type": "wokwi-led",

"id": "led3",

"top": 121.2,

"left": 339.8,

"rotate": 90,

"attrs": { "color": "limegreen" }

},

{ "type": "wokwi-potentiometer", "id": "pot1", "top": 229.1, "left": -125, "attrs": {} },

{ "type": "wokwi-ili9341", "id": "lcd1", "top": 10.4, "left": 422.1, "attrs": {} },

{

"type": "wokwi-pushbutton",

"id": "btn1",

"top": 198.2,

"left": 326.4,

"attrs": { "color": "green", "xray": "1" }

}

],

"connections": [

[ "$serialMonitor:TX", "nucleo:PA3", "", [] ],

[ "$serialMonitor:RX", "nucleo:PA2", "", [] ],

[ "led1:C", "nucleo:GND.9", "black", [ "h0" ] ],

[ "led2:C", "nucleo:GND.9", "black", [ "h-38.4", "v10" ] ],

[ "led3:C", "nucleo:GND.9", "black", [ "h0" ] ],

[ "led1:A", "nucleo:D7", "red", [ "h-67.2", "v220.8" ] ],

[ "led2:A", "nucleo:D4", "yellow", [ "h-57.6", "v192" ] ],

[ "led3:A", "nucleo:D2", "green", [ "h-45.2", "v146.4" ] ],

[ "pot1:VCC", "nucleo:5V.2", "red", [ "v28.8", "h104.8", "v-115.2" ] ],

[ "pot1:GND", "nucleo:GND.5", "black", [ "v48", "h115.2", "v-124.8", "h-28.8" ] ],

[ "lcd1:MISO", "nucleo:D12", "blue", [ "v76.8", "h-403.21", "v-192" ] ],

[ "lcd1:SCK", "nucleo:D13", "violet", [ "v57.6", "h-364.81", "v-182.4" ] ],

[ "lcd1:MOSI", "nucleo:D11", "green", [ "v38.4", "h-336.01", "v-144" ] ],

[ "lcd1:D/C", "nucleo:D0", "magenta", [ "v28.8", "h-288" ] ],

[ "lcd1:CS", "nucleo:D1", "green", [ "v0" ] ],

[ "lcd1:LED", "nucleo:D8", "green", [ "v96", "h-336.01", "v-172.8" ] ],

[ "pot1:SIG", "nucleo:A4", "green", [ "v0" ] ],

[ "btn1:1.l", "nucleo:D9", "green", [ "h0" ] ],

[ "btn1:2.l", "nucleo:GND.9", "green", [ "h-19.2", "v-67" ] ]

],

"dependencies": {}

}

Изображение выглядит как текст, снимок экрана, диаграмма, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 9 – итоговый вид схемы

**Реализация программы**

Класс CarsController реализует управление светофором для транспорта. Управляется с помощью трёх состояний: RED, YELLOW, GREEN.

Листинг 5 - CarsController.h

#ifndef CARS\_CONTROLLER\_H

#define CARS\_CONTROLLER\_H

#include "Include/LED.h"

#define RED\_LED\_PIN D7

#define YELLOW\_LED\_PIN D4

#define GREEN\_LED\_PIN D2

enum class CarsControllerState {RED, YELLOW, GREEN};

class CarsController

{

private:

CarsControllerState state;

public:

LED red = LED(RED\_LED\_PIN);

LED yellow = LED(YELLOW\_LED\_PIN);

LED green = LED(GREEN\_LED\_PIN);

CarsController();

void setState(CarsControllerState \_state);

CarsControllerState getState();

void apply();

};

#endif // CARS\_CONTROLLER\_H

Листинг 6 - CarsController.cpp

#include "Include/CarsController.h"

CarsController::CarsController()

{

state = CarsControllerState::RED;

}

void CarsController::setState(CarsControllerState \_state)

{

state = \_state;

red.Value = (state == CarsControllerState::RED) ? HIGH : LOW;

yellow.Value = (state == CarsControllerState::YELLOW) ? HIGH : LOW;

green.Value = (state == CarsControllerState::GREEN) ? HIGH : LOW;

}

CarsControllerState CarsController::getState()

{

return state;

}

void CarsController::apply()

{

red.apply(); yellow.apply(); green.apply();

}

Листинг 7 – LED.h

#ifndef LED\_H

#define LED\_H

#include <Arduino.h>

class LED

{

public:

uint32\_t Pin;

uint32\_t Value;

LED(uint32\_t pin);

void enable();

void disable();

void toggle();

void apply();

};

#endif // LED\_H

Листинг 8 – LED.cpp

#include "Include/LED.h"

LED::LED(uint32\_t pin)

{

this->Pin = pin;

this->Value = LOW;

}

void LED::enable()

{

this->Value = HIGH;

}

void LED::disable()

{

this->Value = LOW;

}

void LED::toggle()

{

++this->Value %= 2;

}

void LED::apply()

{

pinMode(Pin, OUTPUT);

digitalWrite(Pin, Value);

}

Класс ZebraController управляет TFT-дисплеем. Пешеходный переход (зебра) имеет два состояния: красный – рисует красного человечка, зеленый – поочерёдно рисует первого и второго зелёного человечка.

Изображения были преобразованы в формат массива байт, но по смыслу это массив бит – мы не храним данные цвета для каждого пиксела, только закрашен / не закрашен.

Каждое изображение имеет разрешение 12х16 и масштабируется в 7 раз, то есть рисование происходит не пикселами, а квадратами 7х7.

При большем масштабе или разрешении появляются проблемы с производительностью.

Для работы с SPI дисплеем нужно установить Adafruit GFX и Adafruit ILI9341 библиотеки.

Листинг 9 - ZebraController.h

#ifndef ZEBRA\_CONTROLLER\_H

#define ZEBRA\_CONTROLLER\_H

#include <Arduino.h>

#include "SPI.h"

#include "Adafruit\_GFX.h"

#include "Adafruit\_ILI9341.h"

#define TFT\_CS D1

#define TFT\_DC D0

#define TFT\_LED D8

#define IMAGE\_WIDTH 12

#define IMAGE\_HEIGHT 16

#define SCALE 7

enum class ZebraControllerState {RED, GREEN};

class ZebraController

{

private:

ZebraControllerState state;

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC);

const unsigned char\* image;

uint16\_t imageColor;

public:

ZebraController();

void setState(ZebraControllerState \_state);

void apply();

};

#endif // ZEBRA\_CONTROLLER\_H

Листинг 10 – ZebraController.cpp

#include "Include/ZebraController.h"

#include "Adafruit\_SPITFT.h"

const unsigned char green\_dude [] PROGMEM = {

0x06, 0x00, 0x0f, 0x00, 0x0f, 0x00, 0x0f, 0x00, 0x03, 0x00, 0x03, 0x80, 0x07, 0xc0, 0x1f, 0x60,

0x33, 0x20, 0x63, 0x20, 0x0f, 0xa0, 0x1d, 0x80, 0x18, 0xc0, 0x30, 0xc0, 0x30, 0xe0, 0x60, 0x60

};

const unsigned char red\_dude [] PROGMEM = {

0x00, 0x00, 0x06, 0x00, 0x06, 0x00, 0x3f, 0xc0, 0x3f, 0xc0, 0x2f, 0x40, 0x2f, 0x40, 0x2f, 0x40,

0x2f, 0x40, 0x0f, 0x00, 0x19, 0x80, 0x19, 0x80, 0x19, 0x80, 0x19, 0x80, 0x19, 0x80, 0x30, 0xc0

};

const unsigned char green\_dude\_anim [] PROGMEM = {

0x06, 0x00, 0x0f, 0x00, 0x0f, 0x00, 0x0f, 0x00, 0x03, 0x00, 0x03, 0x00, 0x07, 0x80, 0x07, 0x80,

0x0f, 0x80, 0x1b, 0xc0, 0x13, 0x60, 0x07, 0x00, 0x05, 0x80, 0x04, 0x80, 0x04, 0x80, 0x0d, 0x80

};

ZebraController::ZebraController()

{

state = ZebraControllerState::RED;

tft.begin();

}

void ZebraController::setState(ZebraControllerState \_state)

{

if (\_state == ZebraControllerState::RED)

{

image = red\_dude;

imageColor = ILI9341\_RED;

}

else if (\_state == ZebraControllerState::GREEN)

{

if (image == green\_dude)

{

image = green\_dude\_anim;

}

else

{

image = green\_dude;

}

imageColor = ILI9341\_GREEN;

}

state = \_state;

}

void ZebraController::apply()

{

uint8\_t bytesPerRow = (IMAGE\_WIDTH + 7) / 8;

// Вычисляем смещение для центрирования изображения

uint16\_t xOffset = (tft.width() - IMAGE\_WIDTH \* SCALE) / 2;

uint16\_t yOffset = (tft.height() - IMAGE\_HEIGHT \* SCALE) / 2;

for (int16\_t srcY = 0; srcY < IMAGE\_HEIGHT; srcY++) {

for (int16\_t srcX = 0; srcX < IMAGE\_WIDTH; srcX++) {

// Определяем позицию байта и бита в изображении

uint16\_t byteIndex = srcY \* bytesPerRow + srcX / 8;

uint8\_t bitIndex = 7 - (srcX % 8);

uint8\_t byte = pgm\_read\_byte(&image[byteIndex]);

// Определяем цвет пикселя

if (byte & (1 << bitIndex)) {

// Рисуем масштабированный прямоугольник вместо пикселя

tft.fillRect(

xOffset + srcX \* SCALE,

yOffset + srcY \* SCALE,

SCALE,

SCALE,

imageColor

);

}

else

{

tft.fillRect(

xOffset + srcX \* SCALE,

yOffset + srcY \* SCALE,

SCALE,

SCALE,

ILI9341\_BLACK

);

}

}

}

}

Класс FlagButton слушает события нажатия кнопки. Для этого он регистрирует handleClickCallback.

Важно понимать, что при работе с кнопкой часто наблюдается дребезг – состояние, при котором контакт не стабилен, из-за этого регистрируется много лишних нажатий. Класс регистрирует их все и запоминает их количество, которое можно сбросить.

Листинг 11 – FlagButton.h

#ifndef FLAGBUTTON\_H

#define FLAGBUTTON\_H

#include <Arduino.h>

#define BUTTON\_PIN D9

class FlagButton

{

public:

static inline uint32\_t contactCount;

FlagButton();

uint32\_t reset();

uint32\_t getValue();

};

void handleClickCallback();

#endif // FLAGBUTTON\_H

Листинг 12 – FlagButton.cpp

#include "Include/FlagButton.h"

#include <Arduino.h>

void handleClickCallback()

{

FlagButton::contactCount += 1;

Serial.println("User requested GREEN!");

}

FlagButton::FlagButton()

{

pinMode(BUTTON\_PIN, INPUT\_PULLUP);

attachInterrupt(digitalPinToInterrupt(BUTTON\_PIN), &handleClickCallback, FALLING);

}

uint32\_t FlagButton::reset()

{

uint32\_t temp = contactCount;

contactCount = 0;

return temp;

}

uint32\_t FlagButton::getValue()

{

return contactCount;

}

Клас Knob читает значение регулятора. При тестах выяснилось, что analogRead() конфликтует с SPI дисплеем. То есть, если вызвать analogRead() после tft.begin(), то дисплей перестанет реагировать на команды. Возможно, это проблема только самой симуляции, поскольку в документации Wokwi к потенциометру сказано, что на самом деле Wokwi не умеет работать с аналоговым сигналом и происходит эмуляция.

Поэтому логику программы пришлось временно изменить: она прочитает значение потенциометра только 1 раз в начале, до tft.begin и затем не будет реагировать на изменения регулятора. Для вступления изменений в силу необходимо перезапускать программу каждый раз при изменении регулятора.

Листинг 13 – Knob.h

#ifndef KNOB\_H

#define KNOB\_H

#include <Arduino.h>

#define KNOB\_SIG\_PIN A4

class Knob

{

private:

public:

Knob();

uint32\_t readValue();

};

#endif // KNOB\_H

Листинг 14 – Knob.cpp

#include "Include/Knob.h"

Knob::Knob()

{

pinMode(KNOB\_SIG\_PIN, INPUT);

}

uint32\_t Knob::readValue()

{

//return 400;

return analogRead(KNOB\_SIG\_PIN);

}

Класс Dispatcher реализует основную логику программы.

В России дорожные светофоры работают по определённому циклу: красный, жёлтый, зелёный, снова жёлтый и т. д.

При этом, если светофор оборудован кнопкой вызова, то при нажатии на неё система разрешает перейти через дорогу не в момент нажатия на кнопку, а тогда, когда удобно самой системе в целом. **Не важно, сколько раз пользователь нажмёт на кнопку, система всё равно будет считать это за 1 запрос.** Это совпадает с поведением светофора в реальности и сразу решает проблему с дребезгом.

Можно предположить, что она всегда находится в вышеуказанном цикле, просто пропускает зелёный свет для пешехода, если не было запросов. Именно эту логику реализует класс. Он поочерёдно обрабатывает программу, которая состоит из инструкций. Каждая инструкция состоит из 4-х команд: цвет для пешеходного перехода, цвет для авто, сколько он будет гореть, пропускать ли инструкцию, если кнопка не была нажата. Также с помощью setInstructionDelay можно поменять задержку определённой инструкции, в частности время зелёного света для пешеходов.

Листинг 15 - Dispatcher.h

#ifndef DISPATCHER\_H

#define DISPATCHER\_H

#include "Include/CarsController.h"

#include "Include/ZebraController.h"

#include "Include/FlagButton.h"

struct Instruction

{

ZebraControllerState zebra;

CarsControllerState cars;

uint32\_t delayMillis;

bool flagButtonDependency = true;

};

class Dispatcher

{

private:

uint32\_t stateTicks;

uint32\_t millisPerTick;

int16\_t instructionIndex;

int16\_t executedInstructionIndex;

void apply();

void executeInstruction();

void nextInstruction();

public:

Instruction\* roadProgram;

CarsController\* carsController;

ZebraController\* zebraController;

FlagButton\* flagButton;

size\_t instructionCount;

Dispatcher(Instruction\* \_roadProgram, size\_t \_instructionCount, CarsController\* \_carsController, ZebraController\* \_zebraController, FlagButton\* \_flagButton, uint32\_t \_millisPerTick);

void initialize();

void call();

void setInstructionDelay(size\_t instructionIndex, uint32\_t delayMillis);

};

#endif // DISPATCHER\_H

Листинг 16 - Dispatcher.cpp

#include "Include/Dispatcher.h"

#include <Arduino.h>

Dispatcher::Dispatcher(Instruction \*\_roadProgram, size\_t \_instructionCount, CarsController\* \_carsController, ZebraController\* \_zebraController, FlagButton\* \_flagButton, uint32\_t \_millisPerTick)

{

carsController = \_carsController;

zebraController = \_zebraController;

roadProgram = \_roadProgram;

flagButton = \_flagButton;

instructionCount = \_instructionCount;

millisPerTick = \_millisPerTick;

stateTicks = 0;

instructionIndex = 0;

executedInstructionIndex = -1;

}

void Dispatcher::executeInstruction()

{

Serial.print("Execute Instruction: #");

Serial.print(instructionIndex);

Serial.print(", State ticks: ");

Serial.print(stateTicks);

Serial.print(", ms eval: ");

Serial.print(stateTicks \* millisPerTick);

if (stateTicks \* millisPerTick >= roadProgram[instructionIndex].delayMillis)

{

nextInstruction();

}

if (flagButton->getValue() < 1 && executedInstructionIndex > -1)

{

while (roadProgram[instructionIndex].flagButtonDependency)

{

nextInstruction();

}

}

else if (flagButton->getValue() >= 1 && executedInstructionIndex > -1)

{

if (roadProgram[executedInstructionIndex].flagButtonDependency && !roadProgram[instructionIndex].flagButtonDependency)

{

flagButton->reset();

}

}

carsController->setState(roadProgram[instructionIndex].cars);

zebraController->setState(roadProgram[instructionIndex].zebra);

executedInstructionIndex = instructionIndex;

}

void Dispatcher::initialize()

{

carsController->setState(roadProgram[instructionIndex].cars);

zebraController->setState(roadProgram[instructionIndex].zebra);

carsController->apply();

zebraController->apply();

}

void Dispatcher::apply()

{

carsController->apply();

zebraController->apply();

stateTicks++;

}

void Dispatcher::call()

{

executeInstruction();

apply();

}

void Dispatcher::nextInstruction()

{

++instructionIndex %= instructionCount;

stateTicks = 0;

}

void Dispatcher::setInstructionDelay(size\_t instructionIndex, uint32\_t delayMillis)

{

roadProgram[instructionIndex].delayMillis = delayMillis;

}

В главном файле программы определяется const uint32\_t TACT, который задаёт целевое значение микросекунд для одной итерации главного цикла.

Например, значение 500000 приведёт к тому, что итерация будет пытаться выполниться не чаще, чем раз в полсекунды, то есть экран будет обновляться 2 раза в секунду.

Программа для Dispatcher состоит из 4-х инструкций:

Листинг 17 – инструкции для системы светофора

Instruction\* roadProgram = new Instruction[instructionCount]{

{ZebraControllerState::RED, CarsControllerState::YELLOW, 5000},

{ZebraControllerState::RED, CarsControllerState::GREEN, 30000, .flagButtonDependency=false},

{ZebraControllerState::RED, CarsControllerState::YELLOW, 5000},

{ZebraControllerState::GREEN, CarsControllerState::RED, 30000},

};

Задержка указывается в мс.

Сразу же после задания программы она редактируется с учётом значения с регулятора (мы успели его прочитать до инициализации дисплея, т.к. если читать после, то analogRead будет конфликтовать с SPI):

Листинг 18 – установка значения с регулятора в систему светофора

dispatcher->setInstructionDelay(3, knobValue \* 100);

Листинг 19 – TrafficLight.ino

#include <Arduino.h>

#include "Include/CarsController.h"

#include "Include/ZebraController.h"

#include "Include/Dispatcher.h"

#include "Include/FlagButton.h"

#include "Include/Knob.h"

const uint32\_t TACT = 500000;

uint32\_t debug\_delay;

uint32\_t micros\_buffer;

Dispatcher\* dispatcher;

Knob\* knob;

void setup() {

Serial.begin(115200);

knob = new Knob();

int knobValue = knob->readValue();

CarsController carsController = CarsController();

ZebraController zebraController = ZebraController();

FlagButton flagButton = FlagButton();

size\_t instructionCount = 4;

Instruction\* roadProgram = new Instruction[instructionCount]{

{ZebraControllerState::RED, CarsControllerState::YELLOW, 5000},

{ZebraControllerState::RED, CarsControllerState::GREEN, 30000, .flagButtonDependency=false},

{ZebraControllerState::RED, CarsControllerState::YELLOW, 5000},

{ZebraControllerState::GREEN, CarsControllerState::RED, 30000},

};

dispatcher = new Dispatcher(roadProgram, instructionCount, &carsController, &zebraController, &flagButton, TACT / 1000);

dispatcher->initialize();

dispatcher->setInstructionDelay(3, knobValue \* 100);

Serial.print(" ZEBRA\_GREEN\_DELAY\_ms: ");

Serial.print(knobValue \* 100);

}

void loop() {

micros\_buffer = micros();

dispatcher->call();

debug\_delay = (TACT + micros\_buffer - micros()) / 1000;

delay(debug\_delay);

Serial.print(" debug\_delay\_ms: ");

Serial.println(debug\_delay);

}

**Тестирование**

Зададим небольшое значение на регуляторе.

Изображение выглядит как текст, снимок экрана, диаграмма, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 10 – положение регулятора

Если просто включить симуляцию и ничего не нажимать, то система сразу переключится с инструкции 0 (жёлтый для авто, красный для пешеходов) на инструкцию 1 (зелёный для авто, красный для пешеходов) и будет находиться на этой инструкции, пропуская все остальные.

Листинг 20 – Вывод запуска системы без нажатия на кнопку.

ZEBRA\_GREEN\_DELAY\_ms: 36000Execute Instruction: #0, State ticks: 0, ms eval: 0 debug\_delay\_ms: 346

Execute Instruction: #0, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 13, ms eval: 6500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 14, ms eval: 7000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 15, ms eval: 7500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 16, ms eval: 8000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 17, ms eval: 8500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 18, ms eval: 9000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 19, ms eval: 9500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 20, ms eval: 10000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 21, ms eval: 10500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 22, ms eval: 11000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 23, ms eval: 11500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 24, ms eval: 12000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 25, ms eval: 12500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 26, ms eval: 13000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 27, ms eval: 13500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 28, ms eval: 14000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 29, ms eval: 14500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 30, ms eval: 15000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 31, ms eval: 15500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 32, ms eval: 16000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 33, ms eval: 16500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 34, ms eval: 17000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 35, ms eval: 17500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 36, ms eval: 18000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 37, ms eval: 18500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 38, ms eval: 19000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 39, ms eval: 19500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 40, ms eval: 20000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 41, ms eval: 20500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 42, ms eval: 21000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 43, ms eval: 21500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 44, ms eval: 22000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 45, ms eval: 22500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 46, ms eval: 23000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 47, ms eval: 23500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 48, ms eval: 24000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 49, ms eval: 24500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 50, ms eval: 25000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 51, ms eval: 25500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 52, ms eval: 26000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 53, ms eval: 26500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 54, ms eval: 27000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 55, ms eval: 27500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 56, ms eval: 28000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 57, ms eval: 28500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 58, ms eval: 29000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 59, ms eval: 29500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 60, ms eval: 30000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 13, ms eval: 6500

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 11 – состояние системы без запроса от пользователя

Теперь попробуем снова запустить симуляцию и через пару секунд нажать на кнопку вызова.

Система дождётся удобного момента (когда истечёт время 30 секунд для инструкции 1) и не станет пропускать следующую инструкцию 2 (жёлтый для авто и красный для пешеходов), поскольку увидит, что поступил запрос. После истечения инструкции 2 переключится на инструкцию 3 (зелёный для пешеходов, красный для авто) и будет находиться в ней то количество времени, которое было задано регулятором до начала программы. После этого цикл завершится и снова начнётся инструкция 0 (жёлтый для авто, красный для пешеходов), затем система вернётся в инструкцию 1 (зелёный для авто, красный для пешеходов). После этого будет считаться, что запрос выполнен, и счётчик нажатий кнопки обнулится.

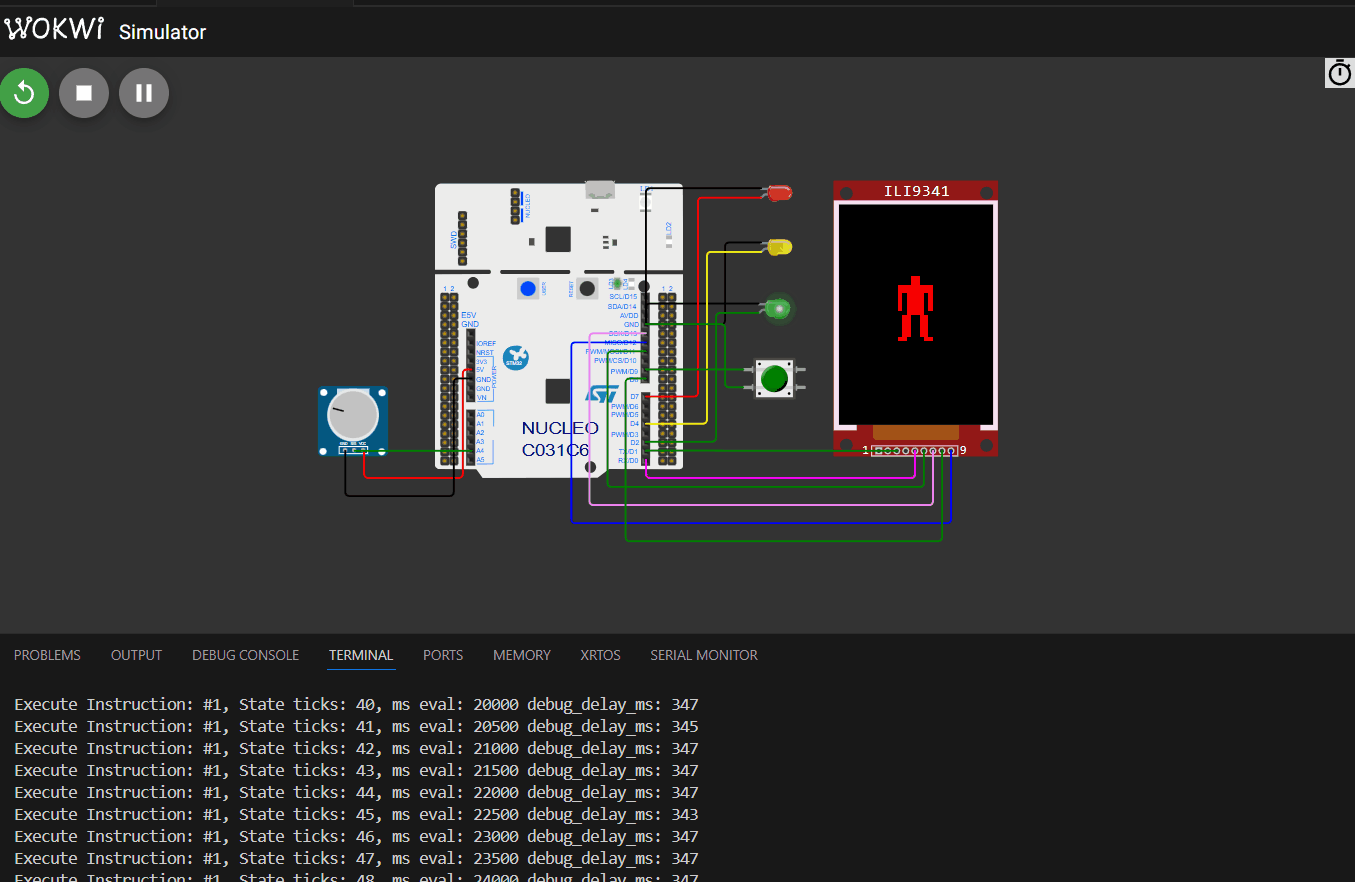


Рис. 12 – реакция системы на нажатие кнопки

Также заметим что кнопка дребезжит, но на стабильность работы системы это никак не влияет.

Листинг 21 – вывод программы для второго запуска

ZEBRA\_GREEN\_DELAY\_ms: 22700Execute Instruction: #0, State ticks: 0, ms eval: 0 debug\_delay\_ms: 346

Execute Instruction: #0, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 8, ms eval: 4000User requested GREEN!

User requested GREEN!

User requested GREEN!

User requested GREEN!

User requested GREEN!

User requested GREEN!

User requested GREEN!

User requested GREEN!

User requested GREEN!

debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 13, ms eval: 6500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 14, ms eval: 7000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 15, ms eval: 7500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 16, ms eval: 8000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 17, ms eval: 8500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 18, ms eval: 9000 debug\_delay\_ms: 345

Execute Instruction: #1, State ticks: 19, ms eval: 9500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 20, ms eval: 10000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 21, ms eval: 10500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 22, ms eval: 11000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 23, ms eval: 11500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 24, ms eval: 12000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 25, ms eval: 12500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 26, ms eval: 13000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 27, ms eval: 13500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 28, ms eval: 14000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 29, ms eval: 14500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 30, ms eval: 15000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 31, ms eval: 15500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 32, ms eval: 16000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 33, ms eval: 16500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 34, ms eval: 17000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 35, ms eval: 17500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 36, ms eval: 18000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 37, ms eval: 18500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 38, ms eval: 19000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 39, ms eval: 19500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 40, ms eval: 20000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 41, ms eval: 20500 debug\_delay\_ms: 345

Execute Instruction: #1, State ticks: 42, ms eval: 21000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 43, ms eval: 21500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 44, ms eval: 22000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 45, ms eval: 22500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 46, ms eval: 23000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 47, ms eval: 23500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 48, ms eval: 24000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 49, ms eval: 24500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 50, ms eval: 25000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 51, ms eval: 25500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 52, ms eval: 26000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 53, ms eval: 26500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 54, ms eval: 27000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 55, ms eval: 27500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 56, ms eval: 28000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 57, ms eval: 28500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 58, ms eval: 29000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 59, ms eval: 29500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 60, ms eval: 30000 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 343

Execute Instruction: #2, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #2, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 348

Execute Instruction: #2, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 343

Execute Instruction: #3, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 343

Execute Instruction: #3, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 13, ms eval: 6500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 14, ms eval: 7000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 15, ms eval: 7500 debug\_delay\_ms: 343

Execute Instruction: #3, State ticks: 16, ms eval: 8000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 17, ms eval: 8500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 18, ms eval: 9000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 19, ms eval: 9500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 20, ms eval: 10000 debug\_delay\_ms: 343

Execute Instruction: #3, State ticks: 21, ms eval: 10500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 22, ms eval: 11000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 23, ms eval: 11500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 24, ms eval: 12000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 25, ms eval: 12500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 26, ms eval: 13000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 27, ms eval: 13500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 28, ms eval: 14000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 29, ms eval: 14500 debug\_delay\_ms: 343

Execute Instruction: #3, State ticks: 30, ms eval: 15000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 31, ms eval: 15500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 32, ms eval: 16000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 33, ms eval: 16500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 34, ms eval: 17000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 35, ms eval: 17500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 36, ms eval: 18000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 37, ms eval: 18500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 38, ms eval: 19000 debug\_delay\_ms: 343

Execute Instruction: #3, State ticks: 39, ms eval: 19500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 40, ms eval: 20000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 41, ms eval: 20500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 42, ms eval: 21000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 43, ms eval: 21500 debug\_delay\_ms: 343

Execute Instruction: #3, State ticks: 44, ms eval: 22000 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 45, ms eval: 22500 debug\_delay\_ms: 347

Execute Instruction: #3, State ticks: 46, ms eval: 23000 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 1, ms eval: 500 debug\_delay\_ms: 348

Execute Instruction: #0, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 343

Execute Instruction: #0, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 347

Execute Instruction: #0, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 13, ms eval: 6500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 14, ms eval: 7000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 15, ms eval: 7500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 16, ms eval: 8000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 17, ms eval: 8500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 18, ms eval: 9000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 19, ms eval: 9500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 20, ms eval: 10000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 21, ms eval: 10500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 22, ms eval: 11000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 23, ms eval: 11500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 24, ms eval: 12000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 25, ms eval: 12500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 26, ms eval: 13000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 27, ms eval: 13500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 28, ms eval: 14000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 29, ms eval: 14500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 30, ms eval: 15000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 31, ms eval: 15500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 32, ms eval: 16000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 33, ms eval: 16500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 34, ms eval: 17000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 35, ms eval: 17500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 36, ms eval: 18000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 37, ms eval: 18500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 38, ms eval: 19000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 39, ms eval: 19500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 40, ms eval: 20000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 41, ms eval: 20500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 42, ms eval: 21000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 43, ms eval: 21500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 44, ms eval: 22000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 45, ms eval: 22500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 46, ms eval: 23000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 47, ms eval: 23500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 48, ms eval: 24000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 49, ms eval: 24500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 50, ms eval: 25000 debug\_delay\_ms: 345

Execute Instruction: #1, State ticks: 51, ms eval: 25500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 52, ms eval: 26000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 53, ms eval: 26500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 54, ms eval: 27000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 55, ms eval: 27500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 56, ms eval: 28000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 57, ms eval: 28500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 58, ms eval: 29000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 59, ms eval: 29500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 60, ms eval: 30000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 13, ms eval: 6500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 14, ms eval: 7000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 15, ms eval: 7500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 16, ms eval: 8000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 17, ms eval: 8500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 18, ms eval: 9000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 19, ms eval: 9500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 20, ms eval: 10000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 21, ms eval: 10500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 22, ms eval: 11000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 23, ms eval: 11500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 24, ms eval: 12000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 25, ms eval: 12500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 26, ms eval: 13000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 27, ms eval: 13500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 28, ms eval: 14000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 29, ms eval: 14500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 30, ms eval: 15000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 31, ms eval: 15500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 32, ms eval: 16000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 33, ms eval: 16500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 34, ms eval: 17000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 35, ms eval: 17500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 36, ms eval: 18000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 37, ms eval: 18500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 38, ms eval: 19000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 39, ms eval: 19500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 40, ms eval: 20000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 41, ms eval: 20500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 42, ms eval: 21000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 43, ms eval: 21500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 44, ms eval: 22000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 45, ms eval: 22500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 46, ms eval: 23000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 47, ms eval: 23500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 48, ms eval: 24000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 49, ms eval: 24500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 50, ms eval: 25000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 51, ms eval: 25500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 52, ms eval: 26000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 53, ms eval: 26500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 54, ms eval: 27000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 55, ms eval: 27500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 56, ms eval: 28000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 57, ms eval: 28500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 58, ms eval: 29000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 59, ms eval: 29500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 60, ms eval: 30000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 345

Execute Instruction: #1, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 13, ms eval: 6500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 14, ms eval: 7000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 15, ms eval: 7500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 16, ms eval: 8000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 17, ms eval: 8500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 18, ms eval: 9000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 19, ms eval: 9500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 20, ms eval: 10000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 21, ms eval: 10500 debug\_delay\_ms: 345

Execute Instruction: #1, State ticks: 22, ms eval: 11000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 23, ms eval: 11500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 24, ms eval: 12000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 25, ms eval: 12500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 26, ms eval: 13000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 27, ms eval: 13500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 28, ms eval: 14000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 29, ms eval: 14500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 30, ms eval: 15000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 31, ms eval: 15500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 32, ms eval: 16000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 33, ms eval: 16500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 34, ms eval: 17000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 35, ms eval: 17500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 36, ms eval: 18000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 37, ms eval: 18500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 38, ms eval: 19000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 39, ms eval: 19500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 40, ms eval: 20000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 41, ms eval: 20500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 42, ms eval: 21000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 43, ms eval: 21500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 44, ms eval: 22000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 45, ms eval: 22500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 46, ms eval: 23000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 47, ms eval: 23500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 48, ms eval: 24000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 49, ms eval: 24500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 50, ms eval: 25000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 51, ms eval: 25500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 52, ms eval: 26000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 53, ms eval: 26500 debug\_delay\_ms: 345

Execute Instruction: #1, State ticks: 54, ms eval: 27000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 55, ms eval: 27500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 56, ms eval: 28000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 57, ms eval: 28500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 58, ms eval: 29000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 59, ms eval: 29500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 60, ms eval: 30000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 1, ms eval: 500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 2, ms eval: 1000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 3, ms eval: 1500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 4, ms eval: 2000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 5, ms eval: 2500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 6, ms eval: 3000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 7, ms eval: 3500 debug\_delay\_ms: 348

Execute Instruction: #1, State ticks: 8, ms eval: 4000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 9, ms eval: 4500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 10, ms eval: 5000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 11, ms eval: 5500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 12, ms eval: 6000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 13, ms eval: 6500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 14, ms eval: 7000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 15, ms eval: 7500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 16, ms eval: 8000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 17, ms eval: 8500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 18, ms eval: 9000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 19, ms eval: 9500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 20, ms eval: 10000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 21, ms eval: 10500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 22, ms eval: 11000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 23, ms eval: 11500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 24, ms eval: 12000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 25, ms eval: 12500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 26, ms eval: 13000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 27, ms eval: 13500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 28, ms eval: 14000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 29, ms eval: 14500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 30, ms eval: 15000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 31, ms eval: 15500 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 32, ms eval: 16000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 33, ms eval: 16500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 34, ms eval: 17000 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 35, ms eval: 17500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 36, ms eval: 18000 debug\_delay\_ms: 343

Execute Instruction: #1, State ticks: 37, ms eval: 18500 debug\_delay\_ms: 347

Execute Instruction: #1, State ticks: 38, ms eval: 19000