WIRELESS PROGRAMMING OF TURTLE ROBOTS BUILT BY STUDENTS

Hristo Dragostinov Hristov
Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences, Sofia, Bulgaria
hristo.dr.hristov@gmail.com

БЕЗЖИЧНО ПРОГРАМИРАНЕ НА РОБОТИ КОСТЕНУРКИ ПОСТРОЕНИ ОТ УЧЕНИЦИ

Abstract

This report presents the steps for building a free and open-source web application for programming of turtle robots used for mathematics and informatics inquiries. The robots are designed to be built by the student who will program them later. Building the robots requires only base, widely available electronic components. The main chip used in the robots is ATmega328P - the same chip which is used in Arduino. The application allows wireless programming of the robots using the Logo language.

Keywords: *Robot; Turtle; Geometry; Programming; Arduino; Logo.*

1. INTRODUCTION

Back in 1971, when computers were not yet widely available at schools and homes, Seymour Papert and Synthia Solomon wrote the article "Twenty Things To Do With A Computer" [1] where they share their vision of students exploring mathematics while programming turtle-robots. This article`s 50th anniversary has been celebrated with a special collection of articles describing good educational practices in various countries including Bulgaria [2].

1.1. Logo

The programming language used in the examples in Papert and Solomon`s article is called Logo[1]. It is also considered to be a computer environment for explorations, a philosophy and even a culture [3]. Logo is the language of choice for programming the turtle-robot through the web application presented in this report.

1.2. At least one robot per child

Nowadays, Papert and Solomon`s goal of *turtles to be cheap enough for every kid to play with one* [1, p. 3] is finally possible. Every child can even have a robot in the classroom and another one at home. The electronic components for a robot are so widely available and affordable that the child can build the robot with its own hands, program it in any language and even create a custom programming environment for it. The creation of one such environment is presented in this report.

[1] Logo Foundation website. Available at: https://el.media.mit.edu/logo-foundation/ (last view: 18-08-2025)

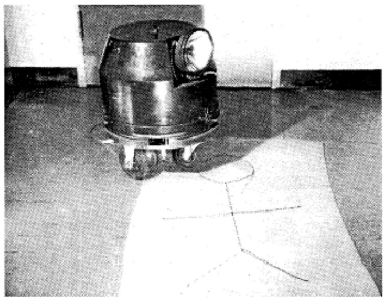

Figure 1: Turtle robot from 1971

1.3. FORWARD and RIGHT

In his book "Mindstorms: children, computers, and powerful ideas", Papert describes the turtle as an "object to think with" [4, p. 9]. So, he decided to use robots, which can draw any geometric shape, by using only two commands, FORWARD and RIGHT. Examples of custom

made robots with some of these capabilities, created nowadays and used in education are EUROPA [5, p. 8] and GeomBot [6, p. 8].

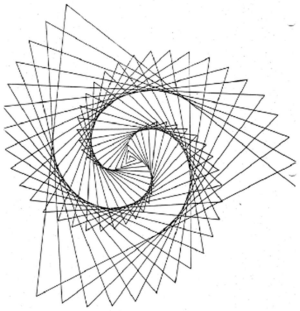

Figure 2: Exploratory challenge for students [1, p. 16]

1.4. The Leo v2 turtle-robot

In 2024, another custom-built robot was designed – Leo v2[2]. Its goal is to be as minimal as possible, but functional enough for Logo. All electronic parts for this robot can be bought from the nearest electronics shop or online. Its body can be made by hand using cardboard, wood or metal – depending on the technologies and tools available at the student`s disposal. All drawings, 3D models, schematics and source code are available on its website and git repository[3].

[2] Leo v2 turtle-robot. Available at: https://hristov-imi-tezo.gitlab.io/vtu-ndmi-2024/ (last view: 18-08-2025)
[3] Leo v2 git repository. Available at: https://gitlab.com/hristov-imi-tezo/vtu-ndmi-2024 (last view: 18-08-2025)

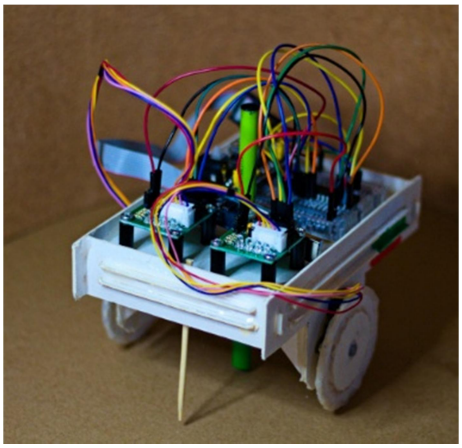

Figure 3: Leo v2 turtle-robot

1.5. Leo v2 programming environment

This report presents the steps for building a webpage, through which the Leo v2 turtle robot can be programmed by students using a computer or a smartphone. Any other robot supporting serial communication through a cable or wirelessly can be programmed with that webpage too, if the robot`s owner is free to program its microchip, which is the case with the Leo v2 robot. It uses the ATmega328P 8-bit AVR microcontroller[4]. Arduino Uno Rev3[5] is another popular choice which has been tested, and has the same microchip built in it.

The steps for programming the robot`s microcontroller are also presented, so it can interpret correctly the serial data sent from the webpage and send a response back to it. An important aspect of this approach is that no additional software should be installed on the computer, except a web browser supporting the Web Serial API[6].

2. MATERIALS AND METHODS

Only free and open-source software[7] is used in every step of this experiment. All created materials are shared in its public Git repository[8] under the GNU General Public License version 3 or later. This allows anyone to design, build and program their own robot, make changes to the design and programs suggested here and share their changes, as long as they give attribution and share their work under a compatible free and open-source software license.

2.1. Steps to reproduce the experiment

1. Download the files from the public Git repository of this experiment[9]
2. Upload **logo-through-serial.ino** (*Appendix C*) to the robot`s microcontroller

[4] ATmega328P Datasheet. Available at: https://content.arduino.cc/assets/ATmega328P_Datasheet.pdf (last view: 18-08-2025)
[5] Arduino Uno Rev3 Documentation. Available at: https://docs.arduino.cc/hardware/uno-rev3/ (last view: 18-08-2025)
[6] Web Serial API. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Web_Serial_API (last view: 18-08-2025)
[7] Free and open-source software. Available at: https://www.gnu.org/philosophy/free-sw.en.html (last view: 18-08-2025)
[8] The public Git repository of this experiment. Available at: https://gitlab.com/hristov-imi-tezo/stemedu-2025 (last view: 18-08-2025)
[9] The public Git repository of this experiment. Available at: https://gitlab.com/hristov-imi-tezo/stemedu-2025 (last view: 18-08-2025)

3. Open **index.html** (*Appendix E*) in a web browser on a computer or smartphone. No internet access is needed. The webpage is fully functional offline.
4. Connect the robot to the computer or smartphone using a USB cable or Bluetooth
5. Click the connect button  on the webpage and choose the correct port, from the dropdown menu, through which the webpage can connect to the robot using the Web Serial API
6. Start sending Logo commands to the robot. The only ones supported in this version are *НАПРЕД* and *НАДЯСНО* – the Bulgarian words for *FORWARD* and *RIGHT* respectively. The webpage translates them to English, before it sends them to the robot, so anyone could translate the commands in another language of choice easily by modifying only **script.js** (*Appendix F*). For the robot to be able to interpret more commands, **logo-through-serial.ino** (*Appendix C*) should also be developed further.

2.2. Hardware needed

1. Computer or smartphone
2. Leo v2 turtle-robot or another robot which uses the ATmega328P microcontroller or Arduino Uno Rev3 and two unipolar stepper motors.
3. Bluetooth module BT06[10], attached to the RX and TX pins of the microcontroller, if wireless communication with the robot is desired

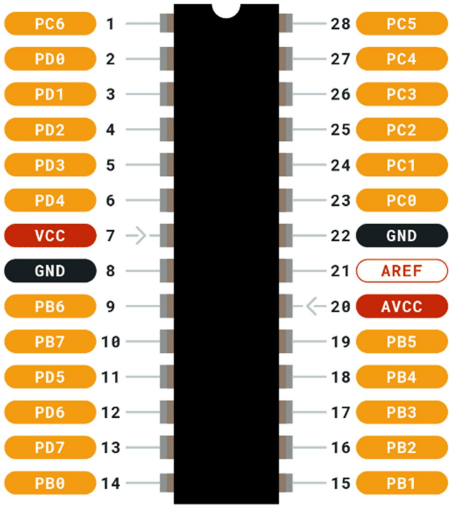

Figure 4: ATmega328P[11]

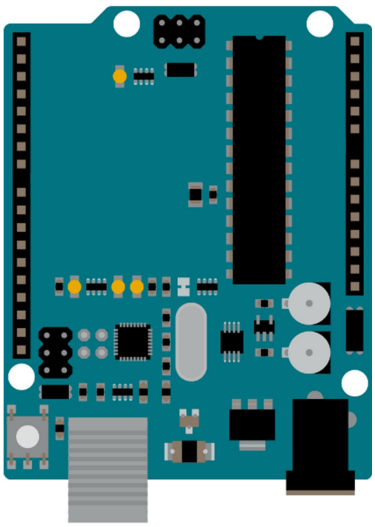

Figure 5: Arduino Uno Rev3[12]

[10] Bluetooth module BT06. Available at: https://www.fabian.com.mt/viewer/42898/pdf.pdf (last view: 18-08-2025)
[11] ATmega328P. Available at: https://docs.arduino.cc/retired/hacking/hardware/ATMEGA328P (last view: 18-08-2025)
[12] Arduino Uno Rev3. Available at: https://docs.arduino.cc/hardwar e/uno-rev3 (last view: 18-08-2025)

2.3. Software needed

This experiment has been conducted under the Arch Linux[13] operating system, but every program used in it is also available for the other GNU+Linux or BSD distributions, MacOS and Windows.

1. **Chromium**[14] or another web browser which supports the Web Serial API. A list of such browsers can be found on the Mozilla Developer website[15].
2. **arduino-cli**[16] to compile the Arduino code to .hex files (*Appendix D*), which can then be uploaded to the ATMega328P microcontroller of the turtle-robot.
3. **avrdude**[17] to upload the .hex files to the ATMega328P microcontroller (*Appendix D*)
4. **Geany**[18] or another text editor

3. RESULTS

3.1. Program for the robot`s microcontroller (Appendices A, B, C, D)

1. **steppermotor.h** (*Appendix A*) This is the header file for handling the stepper motors of the robot. It has two of them – left and right. There are only 3 public methods for the stepper motors (except the contructors) – turnClockwise(), turnCounterclockwise() and rest()

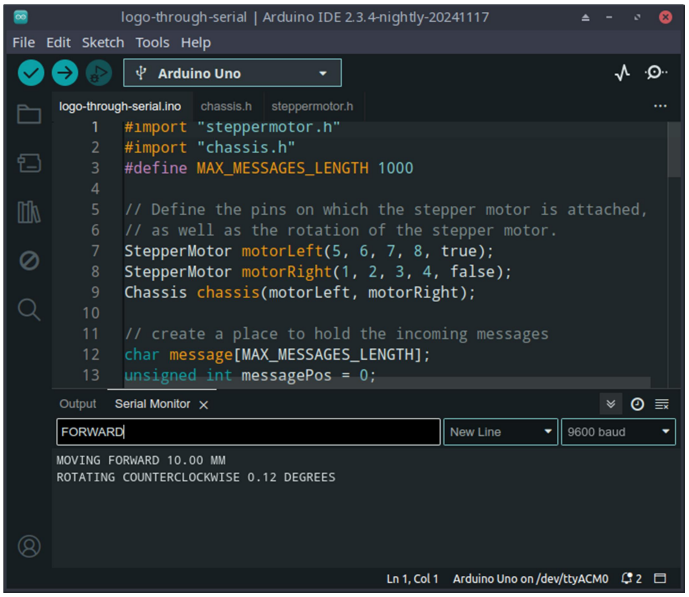

Figure 6: Arduino IDE with its built in Serial monitor used for commands input and output to the Leo v2 turtle-robot

[13] Arch Linux. Available at: https://archlinux.org/ (last view: 18-08-2025)
[14] Chromium web browser. Available at: https://www.chromium.org/Home/ (last view: 18-08-2025)
[15] Web Serial API on the Mozilla website. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Web_Serial_API (last view: 18-08-2025)
[16] arduino-cli. Available at: https://arduino.github.io/arduino-cli (last view: 18-08-2025)
[17] Avrdude. Available at: https://github.com/avrdudes/avrdude (last view: 18-08-2025)
[18] Geany. Available at: https://www.geany.org/ (last view: 18-08-2025)

2. **chassis.h** (*Appendix B*) This is the header file for the chassis, which consists of the two stepper motors and controls them as a unit. This class also has 3 public methods - Figure 6: Arduino IDE with its built in Serial monitor used for commands input and output to the Leo v2 turtle-robot moveForward(float dist), turnRight(float angleDeg), rest(). The methods for moving and rotating can accept a positive or negative decimal fraction or zero.
3. **logo-through-serial.ino** (*Appendix C*) This is the main program, which should be uploaded to the microcontroller. It handles the serial communication, through which the robot receives commands from the student and sends back a response, describing the action it will perform. The language for communication with the robot is English and the commands it supports currently are only FORWARD and RIGHT, followed by a number, on a new line. The number can be an integer or a decimal fraction; positive, negative or zero.

Example:

```
=> FORWARD
=> 10
<= MOVING FORWARD 10.00 MM
=> RIGHT
=> -00.120
<= ROTATING COUNTERCLOCKWISE 0.12 DEGREES
```

4. **upload.sh** (*Appendix D*) This is a Bash[19] shell script for uploading the Arduino code directly to the ATmega328P chip, using a USBasp programmer. The script downloads the MiniCore Arduino library[20], which allows us to compile the source code for the standalone ATmega328P chip, whose clock frequency is 8MHz. The same task can be accomplished by adding the MiniCore library to the Arduino IDE or by using the Arduino Uno Rev3 board, which has a 16MHz quartz crystal allowing us to program it without additional libraries, using only the Arduino IDE.

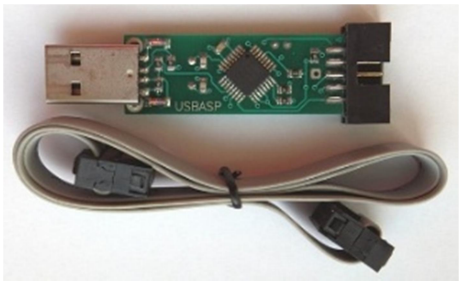

Figure 7: USBasp programmer used for uploading code to the ATmega328P chip[21]

3.2. Webpage (Appendices E, F, G)

1. **index.html** (*Appendix E*) This file defines the layout of the webpage. The programming logic and the styling are extracted in the other two files.

[19] What is Bash? Available at: https://www.gnu.org/software/bash/manual/html_node/What-is-Bash_003f.html (last view: 18-08-2025)
[20] MiniCore library for Arduino. Available at: https://github.com/MCUdude/MiniCore (last view: 18-08-2025)
[21] ATmega328P chip. Available at: https://commons.wikimedia.org/wiki/File:USBasp_programmer.jpg (last view: 18-08-2025)

2. **script.js** (*Appendix F*) This file handles the serialization and deserialization of the commands to and from the microcontroller. It also translates to English, the Bulgarian commands written by the student, translates to pictures the response from the robot, keeps track of the conversation history and validates the input commands, before they are sent to the robot. Any invalid commands are not recorded in history, because otherwise it becomes easily cluttered with wrong commands.

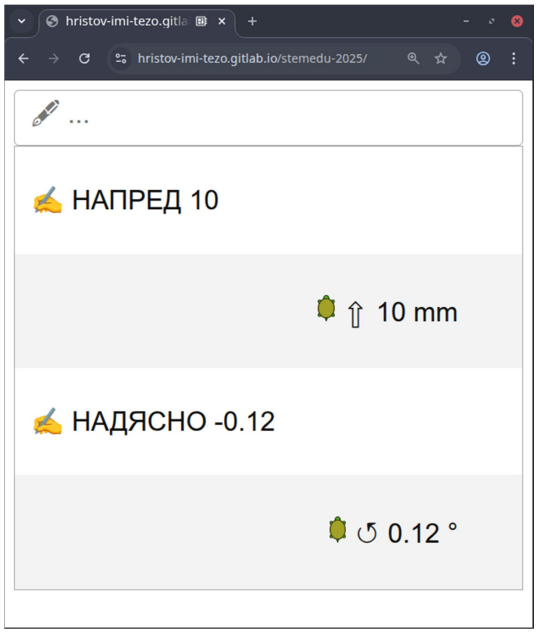

Figure 8: The index.html webpage (Appendix E)

3. **style.css** (*Appendix G*) This handles the styling of the webpage, mainly to make it easily usable on a smartphone screen. An additional CSS library called Picnic CSS[22] is used as a base for the styling. It is downloaded in the repository, so that index.html (Appendix E) looks as good offline as it does online. This allows it to be used in remote areas with no internet connection.

4. DISCUSSION

Several key points were accomplished in this experiment:

4.1. A year-old robot design has been improved to support wireless serial communication through Bluetooth. This opens the possibility for easy communication between the robot and a smartphone. In the case of Android or LineageOS[23] smartphones, the program Serial Bluetooth Terminal[24] can be. In the case of a computer - Arduino IDE`s built-in serial monitor[25] is a popular choice.

[22] Picnic CSS. Available at: https://picnicss.com/ (last view: 18-08-2025)
[23] LineageOS. Available at: https://lineageos.org/ (last view: 18-08-2025)
[24] Serial Bluetooth Terminal. Available at: https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal (last view: 18-08-2025)
[25] Arduino IDE. Available at: https://docs.arduino.cc/software/ide/ (last view: 18-08-2025)

4.2. Connecting wirelessly also has the benefit of safeguarding the computer from eventual wrong wiring in the robot, which would damage the computer`s motherboard otherwise, when connected through USB.

4.3. The robot`s Arduino program **logo-through-serial.ino** (*Appendix C*) can interpret Logo commands in English – FORWARD and RIGHT. More commands can easily be added, supporting more sophisticated Logo syntax like *if* statements, loops, function definitions, recursion and custom procedures. English is the language of choice, because all its letters are ASCII characters, unlike Bulgarian for example. Working with non-ASCII characters often involves deserialization of characters which are more than one byte long, but the robot receives them one byte at a time and this makes debugging a little more confusing, because we can observe the bytes received through the serial communication and the 2-bytes long UTF8 characters there can easily be mistaken for two 1-byte long characters.

4.4. This is true for the lower level, Arduino programming of the robot directly through serial port, but this does not need to be the way for children to program the robot. More user-friendly clients can be developed, taking into account the children`s age, native language, reading skills and experience with programming. One such client has been suggested in the files **index.html** (*Appendix E*), **script.js** (*Appendix F*) and **style.css** (*Appendix G*). It demonstrates that a translation from Bulgarian to English can be accomplished in that front-end, so 1st grade students in Bulgaria who have just learned how to read and write in Bulgarian can program the robot in their native language, avoiding the need to also learn English words, before they can even start programming. The responses from the robot, which are also in English do not need to be displayed like that to the child either. They can be translated into pictures. Having the history of the child-robot conversation is also useful.

4.5. No additional software should be installed on the computer or smartphone, except a browser supporting the Web Serial API. The **index.html** (*Appendix E*) webpage can be opened from a hosted location, like https://hristov-imi-tezo.gitlab.io/stemedu-2025/, which means that also no files need to be downloaded.

4.6. Except when we want to be able to program the robot disconnected from the internet. In which case we should download locally the files for the webpage from the public git repository of this experiment[26]. Once downloaded, it can be opened in a web browser isolated from the internet, because the webpage is self-contained and does not use any shared online resources.

CONCLUSION AND FUTURE WORK

Future steps in the development of this project could be:

- supporting the full Logo syntax in English in the microcontroller
- translating the webpage to support Logo syntax in multiple foreign languages
- adding a menu with pictures representing each Logo command, which would allow children who cannot read yet to be able to program

[26] The public Git repository of this experiment. Available at: https://gitlab.com/hristov-imi-tezo/stemedu-2025 (last view: 18-08-2025)

- adding support for block programming language like Snap![27] or Scratch[28]
- adding the ability to reprogram the ATmega328P chip using Bluetooth, which would allow the creation of a webpage to program the microcontroller in one of its native languages - Arduino, C++, C, AVR Assembly or HEX code through the Bluetooth serial port into the chip. Then the Logo commands can simply become predefined functions written in that native language. The same way, many other programming languages syntax can be predefined - Lisp and Python for example, as well as block programming languages
- then a one-directional translation can be implemented from a block programming language to a text programming language, like it is done in Blocktinu[29] for example.
- a next step would be multidirectional translation, which would allow a child to start programming the robot with blocks programming language, then translate the program to a text programming language and translate it back to the blocks programming language it was originally written in or translate it to another block or text programming language. This way the child can translate the programs into more powerful, but more complex languages as its programming skills improve.

ACKNOWLEDGEMENTS

My deepest gratitude to my mentor Evgenia Sendova for introducing me to Logo and turtle geometry, inspiring me to explore and teach mathematics and informatics this way.

REFERENCES

1. Papert, Seymour A. and Solomon, Cynthia. 1971. Twenty Things to Do with a Computer. DOI: https://doi.org/1721.1/5836
2. Sendova, E. 2021. An Eternal Source of Inspiration or What the Bulgarian Turtle Told Achilles This Time in Gary Stager, Twenty Things to Do with a Computer Forward 50, Constructing Modern Knowledge Press, 63-92, ISBN 978-1-955604-01-7
3. Papert, Seymour A. 1999. What is Logo? And Who Needs It?. Logo philosophy and implementation. Logo Computer Systems Inc., Available at: https://dailypapert.com/what-is-logo-and-who_needs-it/ (last view: 18-08-2025)
4. Papert, Seymour A. 1980. Mindstorms: children, computers, and powerful ideas. ISBN:978-0- 465-04627-0, Available at: https://www.media.mit.edu/publications/mindstorms/ (last view: 18-08-2025)
5. Karalekas, Georgios, Stavros Vologiannidis, and John Kalomiros. 2020. "EUROPA: A Case Study for Teaching Sensors, Data Acquisition and Robotics via a ROS-Based Educational Robot" Sensors 20, no. 9: 2469. DOI: https://doi.org/10.3390/s20092469
6. Baccaglini-Frank, Anna E., George Santi, Agnese Del Zozzo, and Eric Frank. 2020. "Teachers' Perspectives on the Intertwining of Tangible and Digital Modes of Activity with a Drawing Robot for Geometry" Education Sciences 10, no. 12: 387. DOI: https://doi.org/10.3390/educsci10120387

[27] Snap! Available at: https://snap.berkeley.edu/ (last view: 18-08-2025)
[28] Scratch. Available at: https://scratch.mit.edu/ (last view: 18-08-2025)
[29] Blocktinu. Available at: https://blocktinu.com/ (last view: 18-08-2025)

About the Author

Hristo Dragostinov Hristov, PhD student, The Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences (IMI-BAS)

Received: 26-08-2025 Accepted: 19-11-2025 Published: 29-12-2025

Cite as:
Hristov, H. (2025). "Wireless Programming of Turtle Robots Built by Students", Science Series "Innovative STEM Education", volume 07, ISSN: 2683-1333, pp. 158-175, 2025. DOI: https://doi.org/10.55630/STEM.2025.0714

APPENDICES

APPENDIX A

steppermotor.h

```
#define PIN_A_MASK 0b1000
#define PIN_B_MASK 0b0100
#define PIN_C_MASK 0b0010
#define PIN_D_MASK 0b0001

class StepperMotor {
   private:
   int pinA;
   int pinB;
   int pinC;
   int pinD;

// Individual 28BYJ-48 motors rotate in different directions
// (depending on the manufacturer), so this class contains a boolean
// variable indicating the direction of the rotor rotation.
// True means that the motor rotates clockwise when its magnets
// are turned on in the direction A->B->C->D.
// False means that it rotates counterclockwise on the same sequence.
// Clockwise and counterclockwise are defined as
// the direction the rotor rotates, when we place the motor on a flat,
// horizontal surface and look at it from the top.
// In other words, this is the rotation of the wheels when we look at them
// from the side of the robot.

   bool isAbcdClockwiseRotationOrNot;
   int currentMagnetsState = 0b1001; // initial state is 1, 0, 0, 1

   void empowerMagnets() {
      digitalWrite(pinA, currentMagnetsState & PIN_A_MASK);
      digitalWrite(pinB, currentMagnetsState & PIN_B_MASK);
      digitalWrite(pinC, currentMagnetsState & PIN_C_MASK);
      digitalWrite(pinD, currentMagnetsState & PIN_D_MASK);
   }

   void updateMagnetsState(bool isCLockwiseTurn) {
      currentMagnetsState =
      (isCLockwiseTurn == isAbcdClockwiseRotationOrNot) ?
      getNextPos() : getPrevPos();
      empowerMagnets();
   }

   int getNextPos() {
      switch(currentMagnetsState) {
         case 0b1001 : return 0b1100;
         case 0b1100 : return 0b0110;
         case 0b0110 : return 0b0011;
         case 0b0011 : return 0b1001;
      }
   }

   int getPrevPos() {
      switch(currentMagnetsState) {
         case 0b1001 : return 0b0011;
```

```
            case 0b0011 : return 0b0110;
            case 0b0110 : return 0b1100;
            case 0b1100 : return 0b1001;
        }
    }

    public:
// create an empty contructor
    StepperMotor() {}

// create a constructor, which receives the 4 magnet pins as arguments
// and the stepper motor`s direction
    StepperMotor(int a, int b, int c, int d, bool doesRotateClockwise) {
        pinA = a; pinMode(a, OUTPUT);
        pinB = b; pinMode(b, OUTPUT);
        pinC = c; pinMode(c, OUTPUT);
        pinD = d; pinMode(d, OUTPUT);
        isAbcdClockwiseRotationOrNot = doesRotateClockwise;
    }

    void turnClockwise() {
        updateMagnetsState(true);
    }

    void turnCounterclockwise() {
        updateMagnetsState(false);
    }

    void rest() {
        currentMagnetsState = 0b0000;
        empowerMagnets();
    }
};
```

APPENDIX B

chassis.h

```
// Increase this constant for slower motor rotation
// The minimum integer value is 2.
// Faster magnets switching does not give enough time for the rotor to
rotate. #define DELAY_MAGNET_STATES_SWITCH 10
// change for your robot tires, when you have built it
// these values are in millimeters

#define DIAMETER_OF_TYRES 61
#define WHEELS_DISTANCE 115

// do not change these, unless you have built your robot with a different
// motor than 28BYJ-48

#define ROTOR_STEPS_PER_TURN_NO_REDUCTION 32
#define REDUCTION 64

class Chassis {
    private:
    StepperMotor left;
    StepperMotor right;
```

```
    const int ROTOR_STEPS_PER_TURN =
    ROTOR_STEPS_PER_TURN_NO_REDUCTION * REDUCTION;
    const float CIRCUMFERENCE_TIRES = PI * DIAMETER_OF_TYRES;
    const float STEPS_PER_MM = ROTOR_STEPS_PER_TURN / CIRCUMFERENCE_TIRES;
    const float CIRCUMFERENCE_ROBOT_CIRCLE = PI * WHEELS_DISTANCE;
    const float STEPS_PER_DEG = STEPS_PER_MM * CIRCUMFERENCE_ROBOT_CIRCLE
       / 360;

    void moveForwardOnce() {
       left.turnCounterclockwise();
       right.turnClockwise();
       delay(DELAY_MAGNET_STATES_SWITCH);
    }

    void moveBackwardOnce() {
       left.turnClockwise();
       right.turnCounterclockwise();
       delay(DELAY_MAGNET_STATES_SWITCH);
    }

    void turnLeftOnce() {
       left.turnClockwise();
       right.turnClockwise();
       delay(DELAY_MAGNET_STATES_SWITCH);
    }

    void turnRightOnce() {
       left.turnCounterclockwise();
       right.turnCounterclockwise();
       delay(DELAY_MAGNET_STATES_SWITCH);
    }

    public:
       Chassis(StepperMotor m, StepperMotor d) {
          left = m;
          right = d;
       }

       void moveForward(float dist) {
          int n = abs(dist) * STEPS_PER_MM;
          bool isForward = dist > 0;
          for(int i = 0; i < n; i++) {
             isForward ? moveForwardOnce() : moveBackwardOnce();
          }
       }

       void turnRight(float angleDeg) {
          int steps = angleDeg * STEPS_PER_DEG;
          bool isRight = angleDeg > 0;
          for(int i = 0; i < steps; i++) {
             isRight ? turnRightOnce() : turnLeftOnce();
          }
       }

       void rest() {
          left.rest();
          right.rest();
       }
};
```

APPENDIX C

logo-through-serial.ino

```
#import "steppermotor.h"
#import "chassis.h"
#define MAX_MESSAGES_LENGTH 1000

// Define the pins on which the stepper motor is attached,
// as well as the rotation of the stepper motor.
StepperMotor motorLeft(5, 6, 7, 8, true);
StepperMotor motorRight(1, 2, 3, 4, false);
Chassis chassis(motorLeft, motorRight);

// create a place to hold the incoming messages
char message[MAX_MESSAGES_LENGTH];
unsigned int messagePos = 0;
String command = "";
float argument = 0.0;
void setup() {
   Serial.begin(9600);
   delay(1000);
}
void loop() {
// parse the bytes received through the serial port
// and convert them to a string (command) or a number (argument)
   while (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte != '\n' && (messagePos < MAX_MESSAGES_LENGTH - 1)) {
         message[messagePos] = inByte;
         messagePos++;
      } else {
         message[messagePos] = '\0';
         String str = String(message);
         if (str == "FORWARD" || str == "RIGHT") {
            command = str;
         } else {
            argument = str.toFloat();
            if (!argument || command == "") {
               command = "";
               argument = 0;
               Serial.println("?");
            }
         }
         messagePos = 0;
      }
   }
// call the appropriate function based on the received command and argument
// and send a response through the serial port
   if (command != "" && argument) {
      if (command == "FORWARD") {
         Serial.print("MOVING ");
         Serial.print(argument > 0 ? "FORWARD " : "BACKWARD ");
         Serial.print(abs(argument));
         Serial.println(" MM");
         chassis.moveForward(argument);
      } else if (command == "RIGHT") {
         Serial.print("ROTATING ");
         Serial.print(argument > 0 ? "CLOCKWISE " : "COUNTERCLOCKWISE ");
         Serial.print(abs(argument));
```

```
            Serial.println(" DEGREES");
            chassis.turnRight(argument);
        }
        command = "";
        argument = 0;
    }
}
```

APPENDIX D

upload.sh

```
# set the names of the microcontroller and its programmer
PROGRAMMED=atmega328p
PROGRAMMER=usbasp

# get the filename of the first .ino file in this folder
FILENAME=$(ls ./*.ino | head -1)

# compile
arduino-cli cache clean
arduino-cli config add board_manager.additional_urls \
https://mcudude.github.io/MiniCore/package_MCUdude_MiniCore_index.json
arduino-cli core update-index
TEMPDIR=$(mktemp -d)
echo $TEMPDIR
mkdir $TEMPDIR/in
cp $FILENAME $TEMPDIR/in/in.ino
cp *.h $TEMPDIR/in/
arduino-cli compile --clean --fqbn MiniCore:avr:328:clock=8MHz_internal \
--output-dir "$TEMPDIR/out" $TEMPDIR/in

# upload
avrdude -c ${PROGRAMMER} -p ${PROGRAMMED} -b 19200 -U
flash:w:"$TEMPDIR/out/in.ino.hex"
```

APPENDIX E

index.html

```
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="stylesheet" href="lib/picnic/picnic.css">
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <div class="flex" style="height:100vh">
            <button class="pseudo half" id="connectSerialBtn"
                onclick="connectSerial(this)" >
                <img src="img/bluetooth-icon.svg"><img src="img/usb-icon.svg">
            </button>
        </div>
```

```
      <div id="communicationContainer" class="flex" style="display:none">
         <input type="text" id="commandInput"
            onkeypress="onInputKeyPress(event)" placeholder="ד...." >
         <table>
            <tbody id="communicationHistory"></tbody>
         </table>
      </div>
      <script src="script.js"></script>
   </body>
</html>
```

APPENDIX F

script.js

```
// used:
// info: https://developer.chrome.com/docs/capabilities/serial
// info: https://developer.mozilla.org/en-US/docs/Web/API/Web_Serial_API
// example: https://serialmonitor.org/, uses ReactJS, unlike this project
//  example:  https://terminal.spacehuhn.com/,  uses  ReactJS,  unlike  this
project

const communicationContainer =
   document.getElementById("communicationContainer");
const communicationHistory =
   document.getElementById("communicationHistory");

if (!("serial" in navigator)) {
   const connectSerialBtn = document.getElementById("connectSerialBtn");
      connectSerialBtn.disabled = true;
      connectSerialBtn.setAttribute("data-tooltip",
         "Web Serial API is not supported in this browser");
}

async function readSerial(serialReader) {
   const translations = {
      "MOVING": "",
      "ROTATING": "",
      "FORWARD": "⇧",
      "BACKWARD": "⇩",
      "CLOCKWISE": "↻",
      "COUNTERCLOCKWISE": "↺",
      "MM": "mm",
      "DEGREES": "°"
   };

// then move on to the messages we actually want to read
let charIntsArr = [];
while (true) {
   const readerResponse = await serialReader.read();
   charIntsArr.push(...readerResponse.value);
   if (charIntsArr.includes(10) || charIntsArr.includes(13)) {
// newline characters - LF or CR
      const str =
         new TextDecoder("utf-8").decode(new Uint8Array(charIntsArr));
      console.log("received: ",str);
```

```
      str.split(/\r?\n/) // split on the newline characters
         .map(line => line.trim()) // remove lead/trailing spaces from each
line
         .filter(line => !!line) // remove the empty lines
         .forEach(line => {
            let translated;
            if (line === "?") translated = "Ꮗ8;
            else {
               const splt = line = line.split(" ").map(str => str.trim());
               translated = splt.map(str => translations[str] ||
                  Number(str)).
               filter(str => !!str).join(" ");
            }
            communicationHistory.innerHTML += `
 <tr><td align="right">
   <img class="turtle-icons" src="img/turtle.svg"> ${translated}
 </td></tr>
 `;
         });
         charIntsArr = [];
      }
   }
}

async function connectSerial(btn) {
   btn.parentElement.remove(); // remove the connections container
   port = await navigator.serial.requestPort();
   await port.open({ baudRate: 9600 });
   serialWriter = port.writable.getWriter();
   readSerial(port.readable.getReader());
   communicationContainer.removeAttribute("style");
}

async function onInputKeyPress(e) {
   if (event.key == "Enter") {
      const value = event.target.value.trim();
      event.target.value = "";
      if (value) {
         const translations = {
            "НАПРЕД": "FORWARD",
            "НАДЯСНО": "RIGHT"
         };
         const filtered = value.toUpperCase().split(" ").map(str =>
            str.trim()).
         filter(str =>
            !!str  &&  // is not empty string, which would be the double
spaces
                     // and is a number, but not 0
            (translations.hasOwnProperty(str) || (!isNaN(str) &&
               Number(str))));

         // the expected result should be [command, arg]
         if (filtered.length !== 2 ||
            !translations.hasOwnProperty(filtered[0]) ||
            isNaN(filtered[1]))
            return;

         // simplify the number from trailing/leading zeros
         filtered[1] = Number(filtered[1]).toString();
         communicationHistory.innerHTML +=
            `<tr><td>🖌 ${filtered.join(" ")}</td></tr>`;
```

```
            filtered[0] = translations[filtered[0]];
            console.log("to be sent: ",filtered);
            filtered.forEach(str =>
               serialWriter.write(new TextEncoder("utf-8").
               encode(`${str}\n`)));
         }
      }
}
```

APPENDIX G

style.css

```
.flex {
   margin: 0;
   width: 100%;
   padding: 10px;
}

button {
   height: 100%;
}

img {
   width: 45%;
   height: 100%;
   object-fit: contain;
}

#communicationContainer {
   height: calc(100vh - 50px);
}

#communicationContainer * {
  font-size: calc(40px - 2vw);
}

table:has(> #communicationHistory) {
   height: 100%;
}

#communicationHistory {
   border: 1px solid #aaa;
   height: 100%;
}

.turtle-icons {
   width: auto;
   height: 1em;
}
```