Sense C Coding with Arduino

NeuLog

Sense C Coding with Arduino

1_8

For further information contact info@neulog.com

Contents

Chapter 1 – Control and Robots

1.1 Robots

The world today is a world of embedded computer systems. We find them in media systems, watches, phones, remote control, cars, and many more electronics. A few years ago, we did not see terms such as 'wearable computing' or 'internet of things'.

Everyday a surprising new product or application appears and months later, we cannot realize how we lived without it. Modern systems are based more and more on machine learning and artificial intelligence.

The robotic systems, part of the embedded computer system, perform independent activities like search, manipulation, identification, activation, protection and so on.

Many systems combine a certain kind of artificial intelligence in operating and communication between machines.

The robotic system includes the controller, building components, wheels, gears, motors, sensors, and more.

Each robotic system includes a controller that allows it to operate in accordance with different operating programs. The robot developer writes these programs on a computer and forwards them to the controller.

Building a robotic system creates a challenge to acquire knowledge in various technology areas (electronics, computers, mechanics, electricity, etc.).

There are many types of robots such as arm robots, mobile robots, walking robots and more.

The SENSE robots are a series of robots and "brain" units for study, programming and making robots with wide variety of robot applications.

The sense autonomous is a robot which enables us to program many robot applications and functions such as movement on a line, movement along walls, tracking, AGV (Automatic Guided Vehicle), autonomic car, autonomic guard vehicle, autonomic taxi driver, environment monitoring, car manipulation and more. All these applications are described as exercises in this book.

.

1.2 Control systems

A robot is a computerized control system.

A "Control system" may be defined as a group of components, which can be operated together to control multiple variables, which govern the behavior of the system.

Examples:

- Air-conditioning systems control the temperature in the room.
- A greenhouse control system controls temperature, humidity, light, and irrigation.
- A speed control system maintains a steady motor speed regardless of the changing load on the motor.

A light control system can maintain a steady level of light, regardless of the amount of available sunlight. The control system turns lamps ON or OFF according to the requirements.

Three basic units are in every computerized control system:

1. **Input unit** – the unit that reads the system sensors like temperature, light, distance, touch switch, etc. and feeds information into the control unit.

2. **Control unit** – the "BRAIN" of the control system, which contains the system program in its memory and performs the program instructions and processes the received data.

3. **Output unit** – the unit that operates the system actuators such as motors, lamps, pump, and fan as the results of the inputs and the program "decisions".

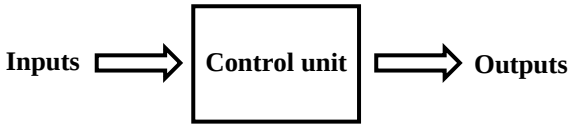


Figure 1-1

The control unit is connected to a computer for programming and downloads a program to the control unit flash memory.

Disconnecting the control unit from the computer and connecting a power source such as a battery to it will create an independent system

1.3 Brain units and NeuLog sensors

Some of the input units can have their own "brain". The NeuLog sensors are such brain units. They send to the control unit, upon request, processed data such as: temperature (^{o}C or ^{o}F), light intensity in Lux, distance in meters, etc.

The output units can also be brain units. For example, units that control the motor speed and direction, lamp intensity, servo motor angle, etc.

These brain units are connected in a chain to the main control unit, which communicates with them through messages.

Every brain unit has an ID number. Every message from the control unit starts with ID number. Only the brain unit with this ID number interprets the message and executes it.

This system construction is the way modern systems are built, and has important advantages:

1. It creates a system with much less wires. The wires go from one module to another and not from all modules to the control unit.

2. This kind of system can easily be changed and expanded, and does not depend on the control units' number of inputs and outputs.

NeuLog sensors (Neuron Logger Sensors) are also brain units. Each sensor includes a tiny computer, which samples, processes and stores the sampled data. Each probe connected to the sensor is pre-calibrated in the factory and no further calibration is required.

The data provided by the sensor is processed digital data. The sensor includes different measurement ranges. Changing the measuring range or type of processing is done simply on the computer screen with NeuLog software.

The sensors are plugged to each other with almost no limitation on the composition and number of sensors in the chain.

NeuLog has over 50 different sensors. Some sensors perform as two to three sensors.

.

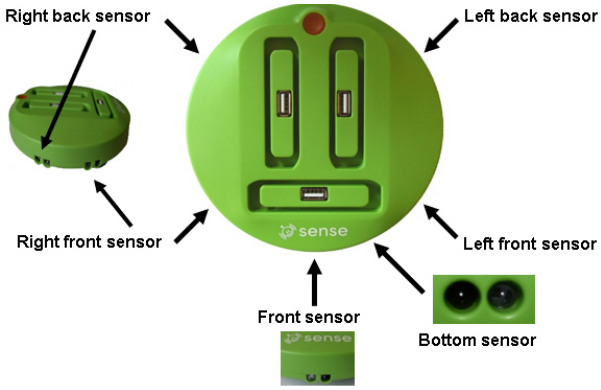
1.4 Sense autonomous

Sense autonomous is a mobile robot for applications such as:

- Movement along black line or white line.
- Movement along walls or in a labyrinth.
- **Autonomous vehicle** such as: AGV, autonomous car, autonomous guard vehicle, autonomous taxi driver, autonomous manipulator.
- Following a moving body holding IR transmitter using tracking module.
- **Environmental monitoring** and measurement robot with NeuLog sensors.

The **sense autonomous** has the following built in:

- Base unit
- 3 connectors for NeuLog sensors or brain units
- 5 IR range sensors
- 1 line sensor
- Pivot wheel
- 2 motors with wheels
- A controller for the base sensors, motors
- A flash memory for the user programs
- USB connector for connection to PC or MAC



In this book, we shall call the sense autonomous in short **SENSE**.

SENSE is programmed as master unit by **RobocklySense**, which can be free downloaded from **www.neulog.com**.

RobocklySense is a special Blockly based program that enables to program all the above applications and more.

You may have the NeuLog battery module BAT-202, which can be plugged directly into one of the SENSE sockets.

When connecting BAT-202 to the SENSE and disconnecting it from the PC, the SENSE becomes an independent robot running on its internal program in its flash memory.

Plugging a coding unit (as described in the following), turns the SENSE automatically to a slave of the coding unit.

Available coding units are:

- WIFI-202 – for programming in Blockly and Python.
- CARM-202 – for programming in C language
- ARD-202 – Arduino board with COM-202 for programming in Arduino C.

1.5 C language

C is a coding language for creating machine programs. These machine programs are fast and work directly with the system hardware components and not through interpreters as the programs above do.

Because its efficiency and simplicity, this language has become popular for developing software for microprocessors and microcontroller embedded systems (or for short, embedded systems).

The book **'C coding with CARM-202**' describes and exercises all about programming in C language with CARM-202.

1.6 CARM-202 C coding unit

The CARM-202 is a C coding unit of the Sense and Neulog series. It is based on the ARM Cortex M3 microcontroller. This microcontroller belongs to the ARM family, which is the leading family of microprocessors and microcontrollers in the world.

CARM-202 is a C language coding unit with 8 switches and 8 LEDs housed in a rigid plastic packaging and colored label.

CARM-202 can be also used as a stand-alone module for ARM microcontroller and for C language programming.

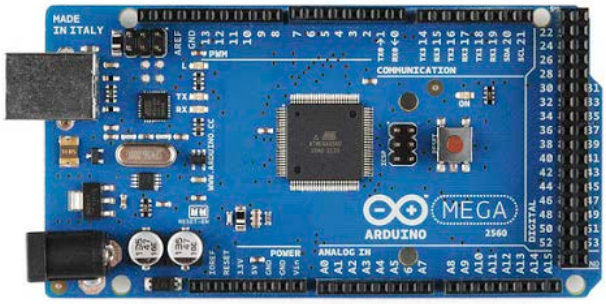
The module has two connectors for communication with NeuLog sensors or with brain I/O units. The module includes flash memory for programs.

The CARM-202 can be powered by the NeuLog battery module or by a USB power source.

1.7 Get an Arduino board and USB cable

In this tutorial, we assume you're using an **Arduino Uno, Arduino Nano, Arduino Mega** or **Diecimila**.

You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example.



1.8 The Arduino board with COM-202

COM-202 is an adapter card for the Arduino board plugged into one of the system's connectors (NeuLog sensor, SENSE robot or brain unit base) and through it to all the system's units.

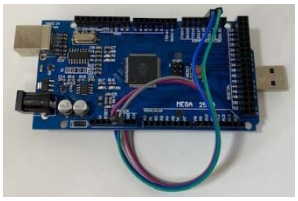
The COM-202 card includes outlet wires for connecting to the communication and power terminals of the Arduino coding cards.

COM-202 comes with software functions that enable communicating with all system units.

Connect the COM-202 to the Arduino board as follows:

- The 5V terminal to the 5V terminal on the Arduino board.
- The GND terminal to the GND terminal on the Arduino board.
- The RX terminal to the TX terminal on the Arduino board.
- The TX terminal to the RX terminal on the Arduino board.

Use a double-sided adhesive tape, to attach the COM-202 to the Arduino board as in the picture.



We shall call this Arduino board with the COM-202 – **ARD-202**.

SES ARD-202 is based on the Arduino Mega 2560.

1.9 Downloading the Arduino Software (IDE)

Get the latest version from the **download page**. When the download finishes, unzip the downloaded file.

1.10 Connecting the board

The Arduino Uno, Mega and Arduino Nano automatically draw power from either the USB connection to the computer or from an external power supply.

Connect the Arduino board to your computer using the USB cable. The green power LED (labelled PWR) should turn on.

1.11 Installing the drivers

1. Installing drivers for the **Arduino Uno** or **Arduino Mega 2560** with Windows 7, Vista, or XP:

   - Plug in your board and wait for Windows to begin its driver installation process. After a few moments, the process will fail, despite its best efforts

   - Click on the **Start Menu**, and open the **Control Panel**.

   - While in the Control Panel, navigate to **System and Security**. Next, click on **System**. Once the System window is up, open the **Device Manager**.

   - Look under **Ports (COM & LPT)**. You should see an open port named **"Arduino UNO (COMxx)"**. If there is no COM & LPT section, look under **"Other Devices"** for **"Unknown Device"**.

   - Right click on the **"Arduino UNO (COmxx)"** port and choose the **"Update Driver Software"** option.

   - Next, choose the **"Browse my computer for Driver software"** option.

   - Finally, navigate to and select the driver file named **"arduino.inf"**, located in the **Drivers** folder of the downloaded Arduino Software (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named **"Arduino UNO.inf"**.

   - Windows will finish up the driver installation from there.

2. Installing drivers for the **Arduino Duemilanove**, **Nano**, or **Diecimila** with Windows7, Vista, or XP:

   When you connect the board, Windows should initiate the driver installation process (if you haven't used the computer with an Arduino board before).

   On Windows Vista, the driver should be automatically downloaded and installed. (Really, it works!)

   On Windows XP, the Add New Hardware wizard will open:

   - When asked "Can Windows connect to Windows Update to search for software?" select "No, not this time", and click next.
   - Select **Install** from a list or specified location (Advanced) and click next.
   - Make sure that **Search for the best driver in these locations** is checked; uncheck **Search removable media**; check **Include this location in the search** and browse to the drivers/FTDI USB Drivers directory of the Arduino distribution (the latest version of the drivers can be found on the **FTDI website**). Click next.
   - The wizard will search for the driver and then tell you that a **"USB Serial Converter"** was found. Click finish.
   - The new hardware wizard will appear again. Go through the same steps and select the same options and location to search. This time, a **"USB Serial Port"** will be found.

   You can check that the drivers have been installed by opening the Windows Device Manager (in the Hardware tab of the System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.

1.12 Launch the Arduino application

Double-click on the Arduino application (arduino.exe) you have previously downloaded.

Note:
If the Arduino Software loads in the wrong language, you can change it in the preferences dialog. See **the Arduino Software (IDE) page** for details.

1.13 Open the blink example

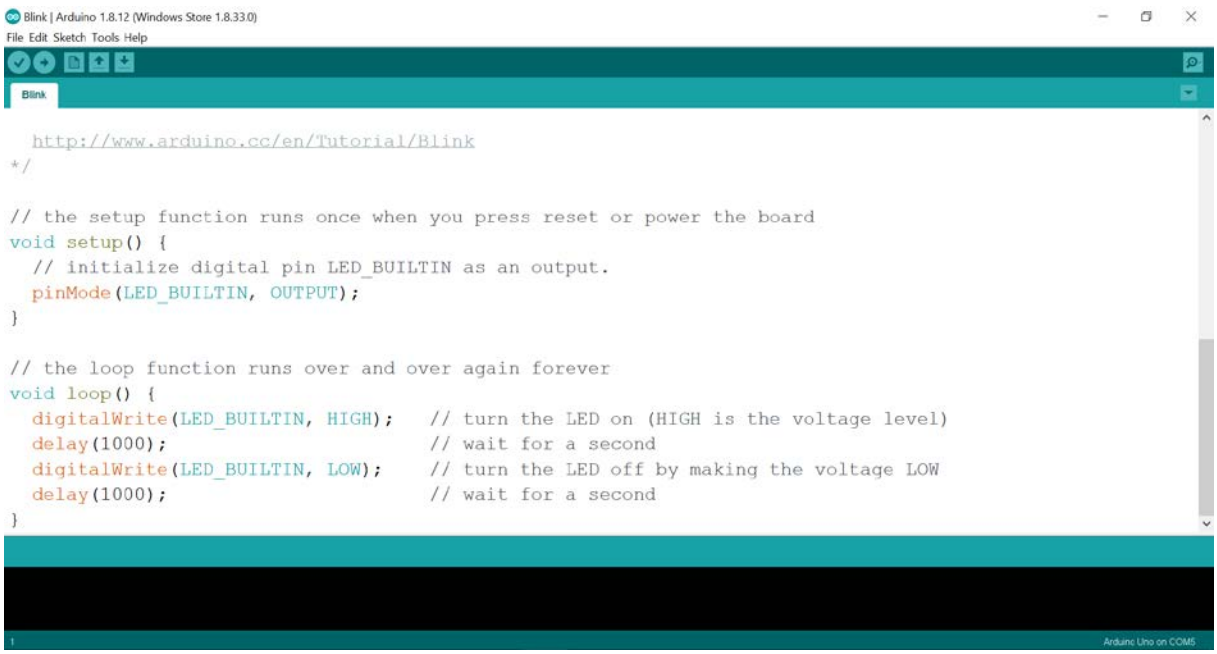
An Arduino program is called **Sketch**.

Double clicking on an Arduino sketch file, runs the Arduino software and open the sketch file in it.

Double click on the **LED blink** Blink example sketch:

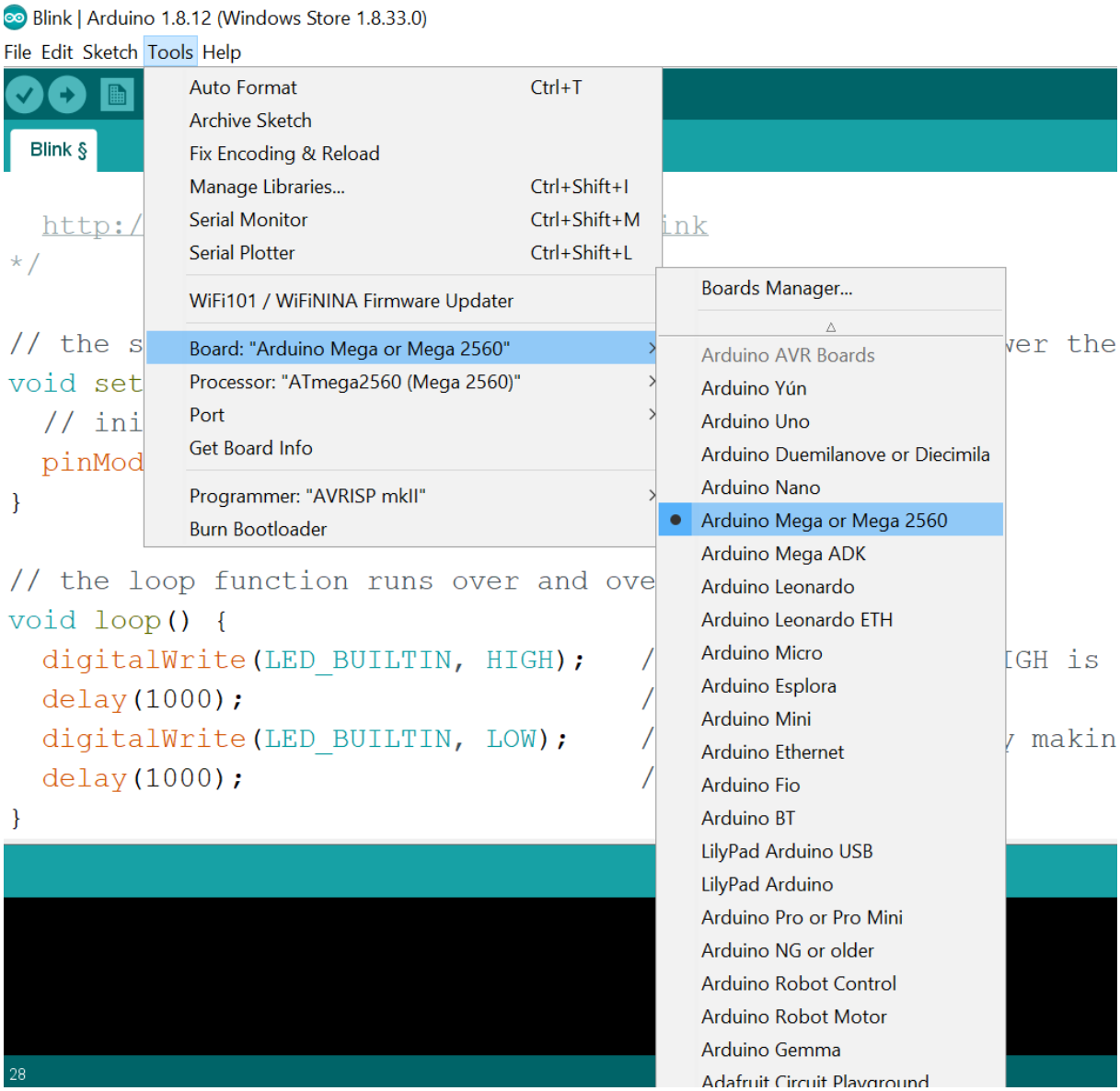
File > Examples > 01.Basics > Blink

Scroll down to view the following screen:



1.14 Selecting your board

You will need to select the entry in the **Tools > Board** menu that corresponds with your Arduino.



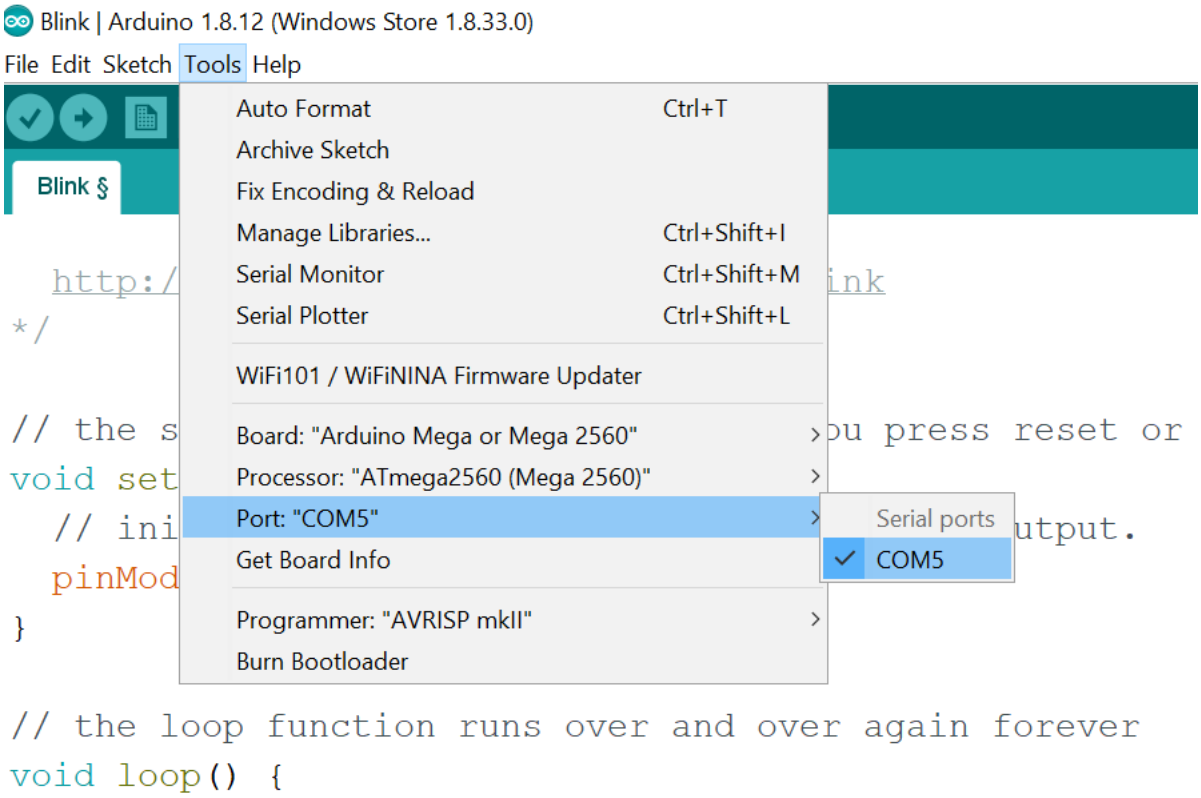
Select the Arduino Mega.

1.15 Selecting your serial port

Select the Arduino board serial device from the **Tools > Serial Port menu**. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).

To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board.

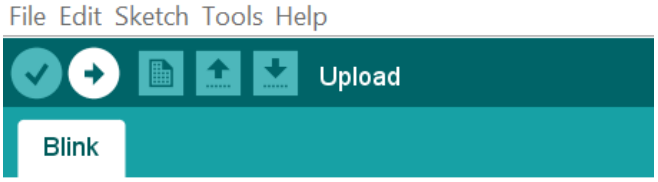
Reconnect the board and select that serial port.



1.16 Uploading the program

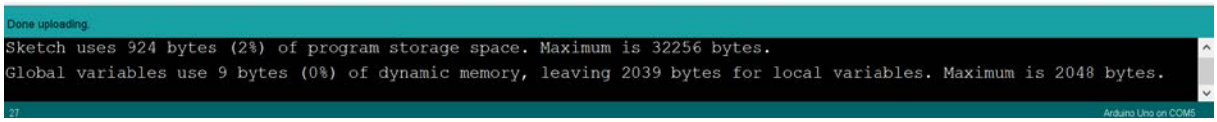
Now, simply click the **Upload** button in the environment. Wait a few seconds – you should see the RX and TX LEDs flashing on the board. If the upload is successful, the message **"Done uploading"** will appear in the status bar.

Note:
If you have an Arduino Mini, NG, or other board, you will need to physically press the reset button on the board immediately before clicking the upload button on the Arduino Software.



The sketch file is compiled and transferred into the flash memory of the Arduino board controller.

A message about the compilation results appears at the bottom of the screen.



A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange).

If it does, congratulations! You've gotten Arduino up-and-running. If you have problems, please check the **https://www.arduino.cc/en/Guide/Troubleshooting**.

1.17 Experiments with SENSE and ARD-202

The library ARD-202 contains the Arduino sketches for the experiments in this book.

You can run and upload the sketches as described in the above sections.

Go to **https://neulog.com/software/** and download **ARD-202.rar**.

Open it and copy the **ARD-202** library into your **Document** library.

Experiment 1.1 – Serial Communication

Objectives:

- Classification of communication methods.
- Serial asynchronous communication.
- UART and USART.
- ASCII code.
- Communication with PC.

Equipment required:

- Computer
- ARD-202 coding unit

Discussion:

1.1.1 Classification of communication methods

In communication between computers, the computers are connected to each other by communication lines. At each stage of the communication, there is a transmitting computer and a receiving computer. The transmitter transmits information through an output port and the receiver receives that information through an input port.

It is possible for a transmitting computer to transmit and then switch into receiving condition and vice versa. No computer can "see" what is happening in the other computer. Computers can only read information, which is placed on their input ports. That is the reason why a part of the transferred information consists of signals concerning the status of the transmitting and the receiving computer, signals such as: "ready to receive", "receive a message", "end of message" etc.

The various communication methods are classified in three basic groups:

a) <u>Synchronous and asynchronous:</u>

In synchronous communication, the computers are connected to a mutual line, which supplies synchronization signals to them both. The synchronization signal enables the computers to know when to transmit and when to expect a message through the communication lines. Each computer, before transmitting a message, awaits the appearance of the synchronization signal, and only then starts transmitting. A computer, which is due to receive a message, awaits the appearance of the synchronization signal and only then collects the information from its input port lines.

In asynchronous communication, we circumvent the use of a synchronization pulse line and a pulse generator. On the information lines, we transmit a start signal at the beginning of each message. The receiving computer awaits the reception of such a signal. After locating it, the receiving computer collects the message, which follows that signal. This method of communication is the most commonly used.

b) Parallel and serial:

In parallel communication, we transmit the information in parallel form. A byte of 8 bits is transmitted through a cable of 8 wires. Each bit is transmitted on a separate wire simultaneously. This method requires a cable with a large number of wires.

In serial communication, we use a small number of wires. The byte is transmitted through one line, bit by bit. The transmitter and receiver must both be synchronized to the same communications frequency.

c) Polling or interrupts:

The problem in communication is in recognizing when the dialogue begins. One of the methods to overcome this obstacle is to determine one of the computers as "MASTER" and the others as "SLAVES". The master always initiates the communication. It turns to the slave and asks whether it has any information to transmit. It waits a certain time to receive a message from the slave. If the slave does not answer within that time, then the master returns to its main program.

The above procedure is performed at pre-determined regular intervals. When the slave has a message to transmit, it waits for the master to turn to it and when this happens, the slave answers by transmitting an opening message. The master reacts and the dialogue takes place. This method is called "communication by polling".

Another method to start a conversation is by interrupts. We use input ports with a strobe line (STB). When one computer wishes to talk to another, it sends a message on its own output port, together with a strobe pulse. An input port collects the message and performs an interrupt request in the receiving computer. The receiver executes the interrupt program, which handles the received message.

This method is quick and convenient although it requires the use of adequate ports and interrupt programs.

Another expression in communication is "handshake". This means that the transmitter of a message awaits an acknowledgement of its reception by the receiver. Without such acknowledgement, the transmitter does not continue with the program.

1.1.2 Serial asynchronous communication

This is the most popular method of communication in microcomputer systems. In this method, the communication line is minimal and may consist of two or three wires only. It is possible to transmit and to receive through telephone wires (with the help of an interface unit called a modem) and even through a wireless connection.

Serial communication is a method in which a byte of 8 bits is translated into a series of serial pulses, zeros and ones, which are transmitted through the communication line. The receiver knows the length of time of each pulse transmitted by the transmitter. In serial asynchronous communication, there is a problem in identifying the start of each byte. The following procedure was therefore determined.

Start bit:

The normal status of the line is "high". Before each byte which is transmitted in a serial form, a '0' bit should be transmitted for the same period of time which is required for the transmission of each of the other bits. This is called "start bit". The receiver identifies the beginning of the transmission of a character by identifying the transition from '1' to '0'.

Data bits:

At the end of the transmission of the start bit, the data bits are transmitted, one after the other. The transmission time of each bit is equal to that of the other bits. Since the receiver knows when the transmission starts, it is able to time the sampling of the data bits in order to overcome the problem of transients.

Parity bit:

Sometimes we use the eighth bit of the data bits as a parity bit, which is used by the receiver to check the accuracy of the data received by it. The value of the bit ('0' or '1') is determined according to the number of 1's in the data byte. There are two ways to determine this: "even parity" and "odd parity". In "even parity", the number of 1's, including the parity bit, should be even. For example, if there are three 1's in a byte, then the transmitter determines the parity bit as '1'. If there are four 1's, then the parity bit will be '0'.

In "odd parity", the number of 1's, including the parity bit, should be odd.

Stop bits:

At the end of each byte, bits of logical 1's are transmitted (usually 2). These bits are used to transfer the line to its normal status for a period of time, which enables the receiver to perform primary processing of the information collected by it, and to resynchronize on the beginning of the transmission of the next character.

The transmission of a single character (58H) will be as follows:

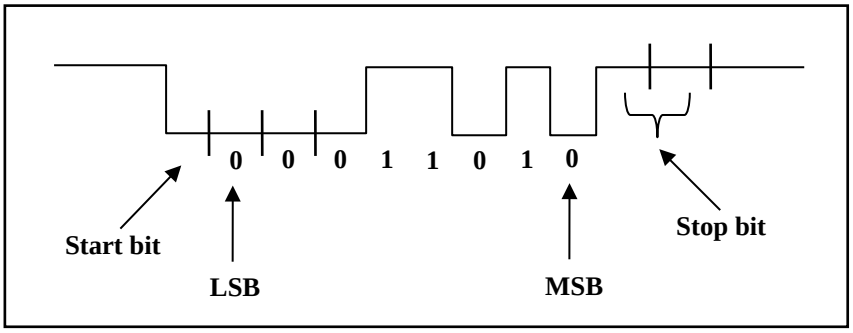


Figure 1-1 Transmission of the character 58H in an asynchronic serial communication

The transmission rate is measured in units of baud, which are bits transmitted in a second. A different transmission is "bits per second", whereby we mean the data bits which are transmitted in one second. For example, if we transmit at a rate of 10 characters per second, the baud rate is 110. Each character requires 11 transmission bits for its transmission (including the start and stop bits). This rate is also equal to 80 bits per second (data bits).

To conclude, asynchronous serial communication is as described in following figure:

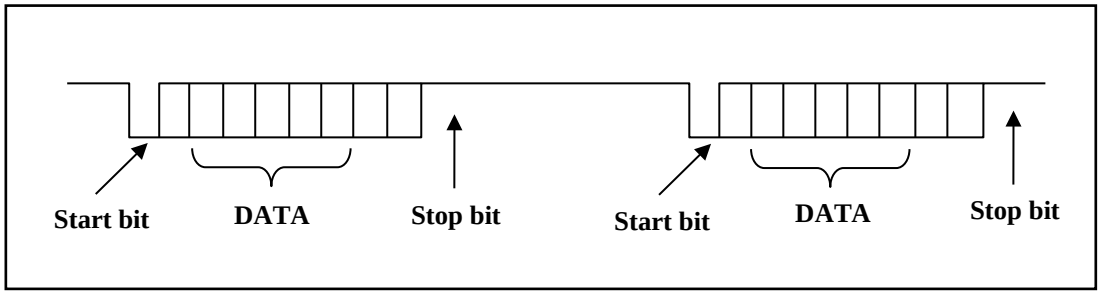


Figure 1-2 Description of the transmission of data in serial communication

The transmitting computer converts a character from its parallel form (as a binary number) into serial form, and then transmits it. The receiving computer translates it back from serial form into parallel form.

It is necessary for the receiver to know the transmission rate and the number of data bits in the transmitted byte (which is not always seven). It also has to know whether the eighth bit indicates even or odd parity or is insignificant, and the number of stop bits.

In communication between computers - one computer transmits and the other receives. Usually, both computers have the capability of transmitting as well as of receiving. Each computer has a TD (Transmit Data) output and a RD (Receive Data) input.

In communication, there are two major forms of connection. One is called: "Full duplex Communication".

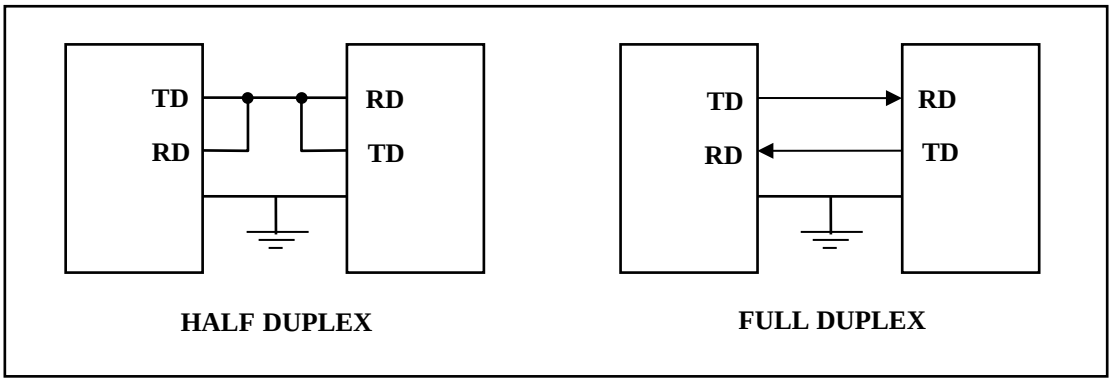


Figure 1-3 The forms of connection in communication

The second form of connection is called: "Half duplex Communication", and it uses only two connecting wires. In half duplex communication, a computer, which changes from receiving into transmitting status must ensure that the other computer has finished transmitting and that it has cleared the line.

Usually, at the input and in the output of a communication line, there are driving components, which enable transmission of the signals over long distances. There are different methods of connection between computers. The most popular are RS232, RS422 and 20 mA current loop.

The process of receiving the serial information and of its conversion into parallel form operates in the following manner: The receiving computer samples the RD input line and awaits the start bit, i.e. the sinking of the line to '0'. As soon as it notices this transition, it waits for a period equaling half a bit time, and then samples the line again. If the line is still '0', that means that the start bit has been received. Now it samples the line at intervals of one bit, according to the number of data bits.

While sampling, the bits are pushed one after the other (LSB is being received first) into a shift register. At the end of the process, the shift register contains the transmitted byte, which is readable in parallel form.

This process is performed with the help of a hardware device called a UART – Universal Asynchronous Receiver Transmitter.

When the UART identifies the START bit, it collects all the DATA bits into a certain buffer register and then creates an interrupt request to tell the CPU that a byte is waiting in the buffer register.

Some UARTs can also work in synchronous mode, which means, starting collecting data only after receiving a certain byte or word. They are called USART.

1.1.3 ASCII code

The ASCII code is a standard, international code used for the exchange of information between input and output units (like the various types of printers, keyboards, external memories) and the computer, as well as between computers. The name ASCII stands for: American Standard Code for Information Interchange.

Each character (letter, digit or other symbol) has been given an agreed upon binary number, by which it is represented in ASCII. For instance, if we want a printer to print the letter A, it must be fed with the binary number 01000001 or 41_{16}.

Following is a table with the various characters and their ASCII code where the ASCII code is expressed in binary, hexadecimal and decimal form.

The first 32 numbers (0-31) are used as special codes for the dialog between the computers like: Start Of Message (SOM), End Of Text (EOT), Carriage Return (CR), Line Feed (LF), etc.

Dec.	Hex.	Binary	Char.
32	20	00100000	SPACE
33	21	00100001	!
34	22	00100010	"
35	23	00100011	#
36	24	00100100	$
37	25	00100101	%
38	26	00100110	&
39	27	00100111	'
40	28	00101000	(
41	29	00101001	)
42	2A	00101010	*
43	2B	00101011	+
44	2C	00101100	,
45	2D	00101101	-
46	2E	00101110	.
47	2F	00101111	/
48	30	00110000	0
49	31	00110001	1
50	32	00110010	2
51	33	00110011	3
52	34	00110100	4
53	35	00110101	5
54	36	00110110	6
55	37	00110111	7
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>
63	3F	00111111	?

Dec.	Hex.	Binary	Char.
64	40	01000000	@
65	41	01000001	A
66	42	01000010	B
67	43	01000011	C
68	44	01000100	D
69	45	01000101	E
70	46	01000110	F
71	47	01000111	G
72	48	01001000	H
73	49	01001001	I
74	4A	01001010	J
75	4B	01001011	K
76	4C	01001100	L
77	4D	01001101	M
78	4E	01001110	N
79	4F	01001111	O
80	50	01010000	P
81	51	01010001	Q
82	52	01010010	R
83	53	01010011	S
84	54	01010100	T
85	55	01010101	U
86	56	01010110	V
87	57	01010111	W
88	58	01011000	X
89	59	01011001	Y
90	5A	01011010	Z
91	5B	01011011	[
92	5C	01011100	↓
93	5D	01011101	←
94	5E	01011110	→
95	5F	01011111	_

Dec.	Hex.	Binary	Char.
96	60	01000000	‘
97	61	01000001	a
98	62	01000010	b
99	63	01000011	c
100	64	01000100	d
101	65	01000101	e
102	66	01000110	f
103	67	01000111	g
104	68	01001000	h
105	69	01001001	i
106	6A	01001010	j
107	6B	01001011	k
108	6C	01001100	l
109	6D	01001101	m
110	6E	01001110	n
111	6F	01001111	o
112	70	01010000	p
113	71	01010001	q
114	72	01010010	r
115	73	01010011	s
116	74	01010100	t
117	75	01010101	u
118	76	01010110	v
119	77	01010111	w
120	78	01011000	x
121	79	01011001	y
122	7A	01011010	z
123	7B	01011011	{
124	7C	01011100	|
125	7D	01011101	}
126	7E	01011110	~
127	7F	01011111	DEL

1.1.4 Communication with PC

The ARD-202 microcontroller has two USARTS. One is for the communication with the PC (USART2) and one for the communication with the SENSE and the NeuLog modules.

The following program waits for character from the serial port and sends it back to the PC followed with the ASCII number of the character received from the PC.

```
/*
  Serial Communication
  Reads characters from serial port.
  Prints the received character in the serial monitor with its ASCII value
  Prints 'Hello' followed by carriage return when the letter H is received.
*/

int Received_Char;

void setup() {
  //Initialize serial for 9600 baud.
  Serial.begin(9600);

  // prints titles
  Serial.println("Type characters on the monitor and send");
  Serial.println("H => prints 'Hello'");
}

void loop() {
  // wait for a serial character, read it and write it:
  while (Serial.available()) {
    Received_Char = Serial.read();
    Serial.write(Received_Char);
    Serial.print("=");
    Serial.print(Received_Char);
    if (Received_Char == 'H') {
      Serial.println();
      Serial.println("Hello");
    }
  }
}
```

The processor does not stop. It fetches an instruction from memory and executes it. This is why a control program runs in endless loop. This is the loop function of Arduino software in the main program.

Some setup instructions are executed once before entering the loop function.

The setup function of the above program includes the serial USART initialization and printing titles and messages.

The instruction **while (Serial.available()) {** causes the loop function instructions to be executed only when a character is received.

The **Received_Char** variable gets the read character and it is printed with its ASCII code.

The **Serial.write(Received_Char)** instruction prints the ASCII character.

Serial.print(Received_Char) instruction prints the ASCII code of the character.

The program prints also '**Hello**', when the letter 'H' is received followed by Carriage Return and Line Feed included in the instruction **Println**.

1.1.5 Arduino programs

Every Arduino program file is saved in a certain library with the same name.

The library and the program names should be without spaces.

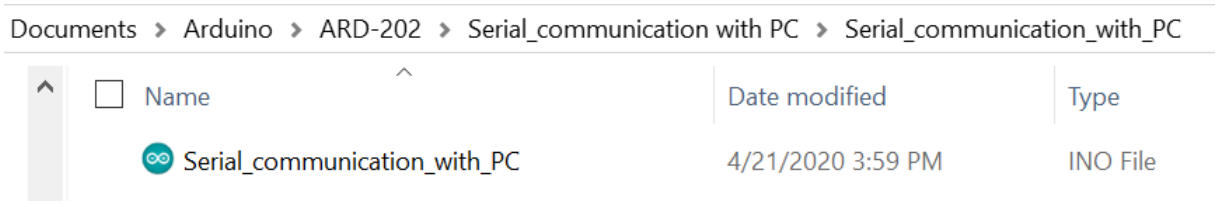
All the example programs in this book are saved in a main library called ARD-202.

Procedure:

1. Connect the Arduino board to your computer using the USB cable. The green power LED (labelled PWR) should turn on.

2. Enter the **Serial_Communiication_with_PC** library in the **ARD-202** main library.

3. Double click on the **Serial_Communiication_with_PC**.



4. Observe the following screen:

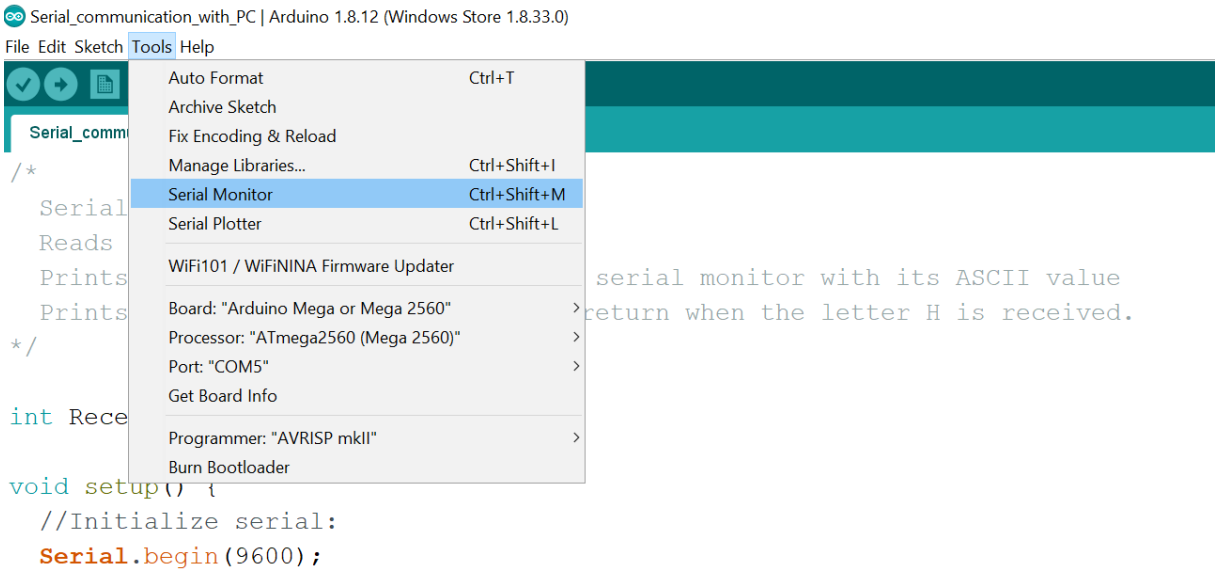
5. Scroll down to view the entire program.

6. Check that you understand all the program instructions according to the description in section 1.1.4.

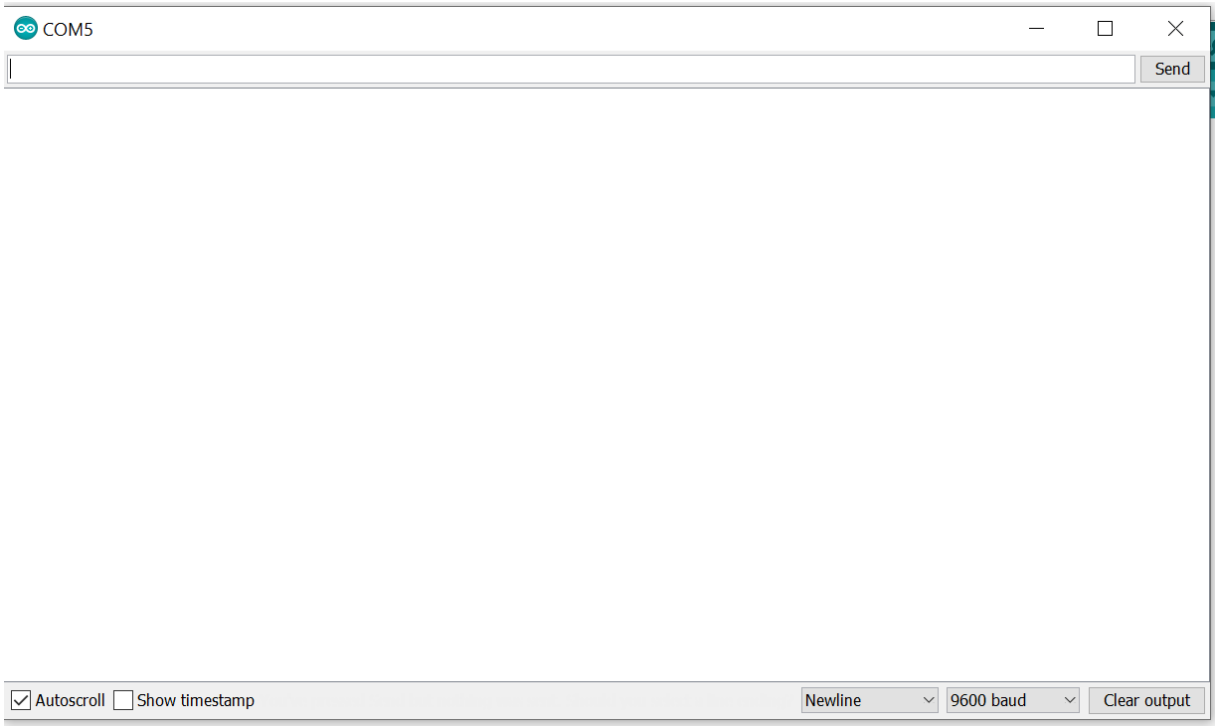
7. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

8. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

9. Choose **Serial Monitor** from the **Tools** menu.

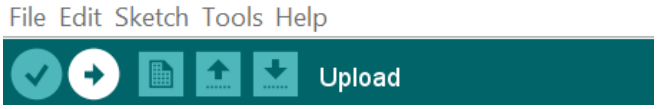


The following serial monitor screen will open:



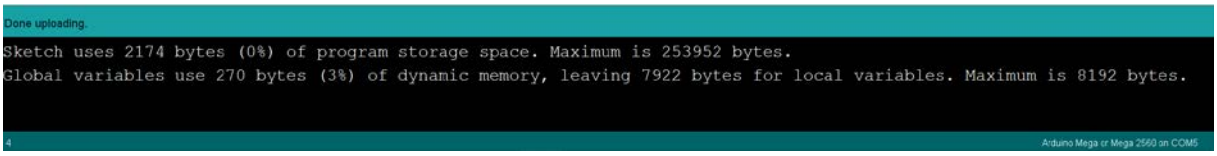
10. Click on the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board. If the upload is successful, the message **"Done uploading"** will appear in the status bar.

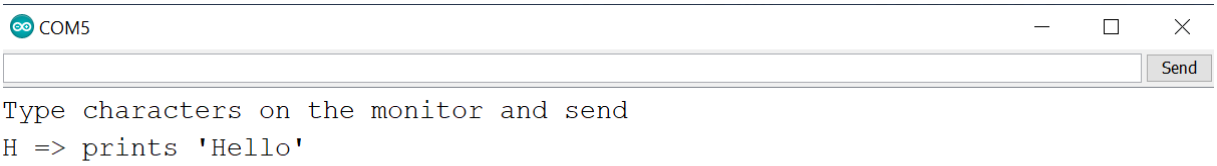


    The sketch file is compiled and transferred to the flash memory of the Arduino board controller.

    A message about the compilation results appears at the bottom of the screen.



    A few seconds after the upload finishes, the following message should appear on the serial monitor.



11. Click on the input field above to see the blinking cursor there.

12. Click on the '**a**' key and click on **Send** on the right.

```
a=97
=10
```

    The letter '**a**' appears followed by its decimal value of its ASCII code (97).

    LF (Line Feed) is also sent and its ASCII code is 10.

13. Click **ABC** and then click **Send**.

14. Click on the '**H**' key and you will get the message '**Hello**' on the monitor.

15. Change the program so it sends the word **'Robot'** when the key **'R'** is pressed.

16. Click the **Upload** button.

17. Press various keys including the **R** key and check the program behavior.

18. Use the '**Switch – Case**' instruction and create a program that sends the name of the day to the terminal when its initial letter is pressed.

19. Click the **Upload** button.

20. Press various keys and check the program behavior.

21. Exit the program.

Experiment 1.2 – Communication with SENSE

Objectives:

- Sending commands to SENSE.
- Forward and backward.
- Changing speed.
- Turning right and left.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 battery module

Discussion:

The previous experiment described how to send characters and strings to the PC.

Talking with the SENSE and all other brain units is also done by sending strings. The brains in these units analyze the strings and execute them. Modern systems are built this way.

The communication between ARD-202, SENSE and brain units is done through USART1.

In order to simplifies the programming, we use a header file **com202.h** with operating functions described below.

The Arduino program requires the header files, which are not in its libraries, so they should be added to the working library.

SetMotor

The following function describes the structure of the instruction to operate the SENSE motors:

SetMotor(robo.type,robo.ID,num,command);

The **robo.type** options are:

Sense, **Robo206**, **RoboEx**, **BrainServo**, **BrainMotor**, **BrainArm**, **IRTrack**

robo.ID is a number from 1 to 9.

The **num** options are:

0 – for all motors
1 – for M1
2 – for M2
3 – for M3

The **command** options are:

off – for stop
cw – for clockwise
ccw – for counter clockwise

For example, the following is a command for the SENSE to move forward:

SetMotor("sense",1,0,cw);

Note:
The distinction between upper and lower case is important.

SetSpeed

We can set the speed of the SENSE motors with the following function:

SetSpeed(robo.type,robo.ID,num,speed);

The **Speed** options are a number between 0 and 255.

The following is a command to set the speed of the SENSE motor number 2 (M2) to 250:

SetSpeed("sense",1,2,250);

Notes:
We use the **SetSpeed** command when we want to change the speed of a motor. The SENSE remembers the last setup speed.

We can set each motor to a different speed.

The following is a program that moves the SENSE forward for 3 seconds, backward for 3 seconds and stops.

```
#include "com202.h"

void setup()
{
  SerialBegin(1);
}

void loop()
{
  SetSpeed("sense",1,0,200);    // set motor speed
  SetMotor("sense",1,0,"cw");   // motors forward
  delay(3000);                  // 3 sec delay
  SetMotor("sense",1,0,"ccw");   // motors backward
  delay(3000);                  // 3 sec delay
  SetMotor("sense",1,0,"off");  // motors off

  exit(0);  //end program
}
```

The function **exit();** creates software reset to the ARD-202 board.

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module into the right socket of the SENSE.

   It does not matter where we plug the modules.

   We do so for the convenience of connecting the communication cable.

   We need the battery module, because it is not recommended to drive the SENSE using the PC USB outlet, and most computers cannot supply enough power to the ARD-202 and the SENSE.

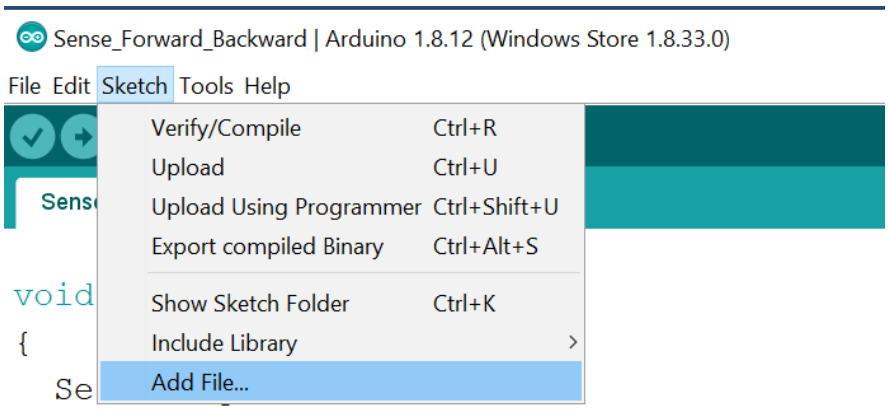
4. Enter the **Sense_Forward_Backward** library in the **ARD-202** main library.

5. Double click on the **Sense_Forward_Backward**.

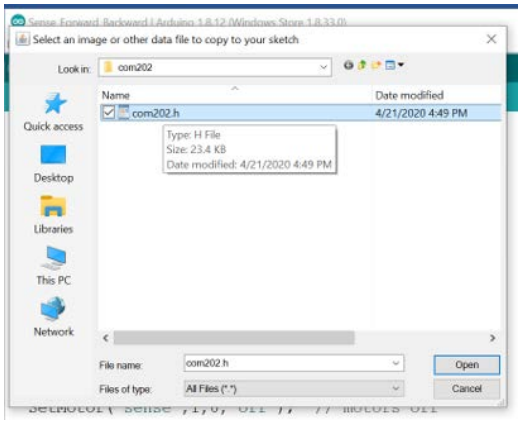
6. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

7. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

8. Choose **Add File...** from the **Sketch** menu.



9. Browse to the com202 library and select the **com202.h** file.



1.2.1 Forward and backward

10. Observe the following screen:

Sense_Forward_Backward | Arduino 1.8.12 (Windows Store 1.8.33.0)

File Edit Sketch Tools Help

Sense_Forward_Backward § | com202.h

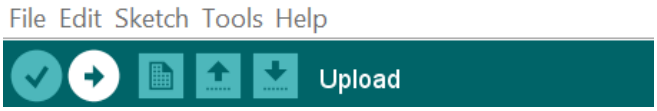
```
#include "com202.h"

void setup()
{
  SerialBegin(1);
}

void loop()
{
  SetSpeed("sense",1,0,200);     // set motor speed
  SetMotor("sense",1,0,"cw");    // motors forward
  delay(3000);                  // 3 sec delay
  SetMotor("sense",1,0,"ccw");    // motors backward
  delay(3000);                  // 3 sec delay
  SetMotor("sense",1,0,"off");   // motors off

  exit(0);  //end program
```

11. Scroll down to view the entire program.

12. Make sure that you understand all the program instructions.

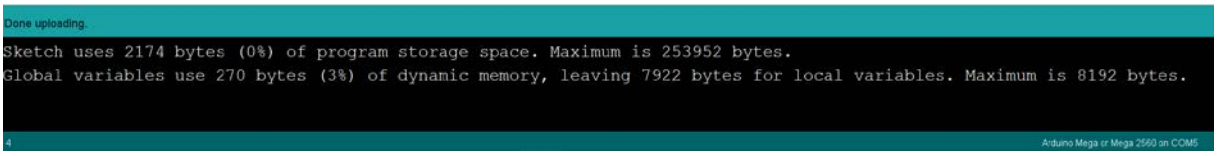
13. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board. If the upload is successful, the message **"Done uploading"** will appear in the status bar.



    The sketch file is compiled and transferred to the flash memory of the Arduino board controller.

    A message about the compilation results appears at the bottom of the screen.



    A few seconds after the upload finishes, the SENSE should move forward for 3 seconds, move backward for 3 seconds and then stop.

    If it does not, press the **RST** (Reset) button on the Arduino board.

14. Disconnect the ARD-202 from the PC.

    The program is stored in the Arduino board flash memory.

15. Put the SENSE on the floor and press the **RST** button on the Arduino board.

    The SENSE should move forward for 3 seconds move backward for 3 seconds and then stop.

1.2.2 Forward, wait and backward

1. Change the program to the following:

```
void loop()
{
  SetSpeed("sense",1,0,200);    // set motor speed

  SetMotor("sense",1,0,"cw");   // motors forward
  delay(3000);                  // 3 sec delay

  SetMotor("sense",1,0,"off");   // motors stop
  delay(1000);                   // 1 sec delay

  SetMotor("sense",1,0,"ccw");   // motors backward
  delay(3000);                  // 3 sec delay

  SetMotor("sense",1,0,"off");  // motors off

  exit(0);  // end program
}
```

   It is always recommended to add a short delay before changing the motor's direction.

2. Click the **Upload** button.

   Wait for the compilation results message and the **"Done uploading"** message.

   A few seconds after the upload finishes, the SENSE should move forward for 3 seconds, wait 1 second, move backward for 3 seconds and stop.

   If it does not, press the **RST** button on the Arduino board.

1.2.3 Turning left and right

For turning, we have to address each motor separately.

We have three options:

Deviating – rotating each motor in a different speed.
Turning – stopping one motor and rotating the other one.
Rotating – rotating one motor forward and the other motor backward.

1. Change the program to the following:

```
void loop()
{
  SetSpeed("sense",1,0,200);    // set motor speed

  SetMotor("sense",1,2,"cw");   // turn left
  SetMotor("sense",1,1,"off");
  delay(3000);              // 3 sec delay

  SetMotor("sense",1,1,"cw");   // turn right
  SetMotor("sense",1,2,"off");
  delay(3000);              // 3 sec delay

  SetMotor("sense",1,0,"off");  // motors off

  exit(0);  // end program
}
```

2. Click the **Upload** button.

   Wait for the compilation results message and the **"Done uploading"** message.

   Hold the SENSE in your hand and press the **RST** button on the Arduino board.

   The right motor should rotate forward for 3 seconds and then the left motor should rotate for 3 seconds.

3. Disconnect the ARD-202 from the PC and place the SENSE on the floor.

4. Press the **RST** button on the Arduino board.

   The SENSE should turn left for 3 seconds, turn right for 3 seconds and then stop.

5. Change the delay time to allow turns of 90º.

1.2.4 Rotating left and right

1. Change the program to the following:

```
void loop()
{
  SetSpeed("sense",1,0,200);    // set motor speed

   SetMotor("sense",1,2,"cw");   // rotate left
   SetMotor("sense",1,1,"ccw");
  delay(3000);              // 3 sec delay

   SetMotor("sense",1,1,"cw");   // rotate right
   SetMotor("sense",1,2,"ccw");
  delay(3000);              // 3 sec delay

   SetMotor("sense",1,0,"off");  // motors off

  exit(0);  // end program
}
```

2. Click the **Upload** button.

   Wait for the compilation results message and the **"Done uploading"** message.

   Hold the SENSE in your hand and press the **RST** button on the Arduino board.

   The right motor should rotate forward and the left motor should rotate backward for 3 seconds and then the left motor should rotate forward and the right motor should rotate backward for 3 seconds.

3. Disconnect the ARD-202 from the PC and place the SENSE on the floor.

4. Press the **RST** button on the Arduino board.

   The SENSE should rotate left for 3 seconds, rotates right for 3 seconds and then stop.

5. Change the delay time to allow turns of 90^{o}.

1.2.5 Deviating left and right

1. Change the program to the following:

```
void loop()
{
 SetSpeed("sense",1,2,200);    // set motor 2 fast speed
 SetSpeed("sense",1,1,100);    // set motor 1 slow speed
  SetMotor("sense",1,0,"cw");   // two motors forward
 delay(3000);              // 3 sec delay

 SetSpeed("sense",1,1,200);    // set motor 1 fast speed
 SetSpeed("sense",1,2,100);    // set motor 2 slow speed
 delay(3000);               // 3 sec delay

  SetMotor("sense",1,0,"off");  // motors off

 exit(0);  // end program
}
```

2. Click the **Upload** button.

   Wait for the compilation results message and the **"Done uploading"** message.

   Hold the SENSE in your hand and press the **RST** button on the Arduino board.

   The right motor should rotate forward fast and the left motor should rotate forward slow for 3 seconds and then the left motor should rotate fast and the right motor should rotate slow for 3 seconds.

3. Disconnect the ARD-202 from the PC and place the SENSE on the floor.

4. Press the **RST** button on the Arduino board.

   The SENSE should deviate left for 3 seconds, deviate right for 3 seconds and then stop.

5. Change the delay time to allow turns of 90º.

1.2.6 Challenge exercises – Moving in a square

Task 1: Make a program that moves the SENSE in a 30x30 cm square until it returns to its original starting point.

Use the **Rotate** instructions for rotating.

Task 2: Make a program that moves the SENSE in a 30x30 cm square until it returns to its original starting point.

Use the **Turn** instructions for rotating.

Experiment 1.3 – Interactive Programs

Objectives:

- Program that reacts to sensors.
- Moving the SENSE to a wall.
- Moving the SENSE to a wall and back.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 battery module

Discussion:

In this experiment, we will move the SENSE to a wall.

We will learn how to read and react to the **Front Range** sensor.

A closed loop system is a control system, which reacts to sensors and switches.

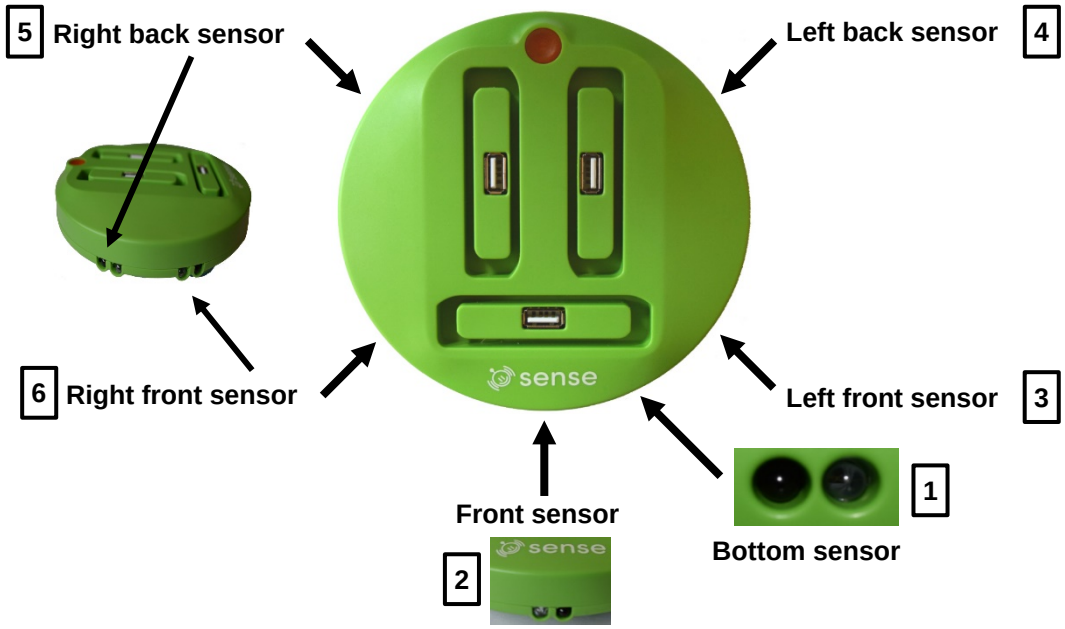
An example for closed loop system is a control system that lights up a lamp when it is dark, and turn it OFF when there is light. This system is automatically adapted to summer time (when the night is short) and to wintertime (when the night is long and starts early).

The program of closed loop system contains decision instructions such as:

'**while**', '**do - while**', '**if – then**'.

1.3.1 The SENSE sensors

The SENSE has 6 sensors. Five range sensors on its perimeter and one line-detector on its bottom.



Each sensor has a number marked on the SENSE as in the above picture.

An interactive program reacts to a value read from the sensor. Before writing a program, we have to know the required sensor number to which the program should react to.

1.3.2 Printing SENSE front sensor value

The following program prints, on the terminal screen, the read values from the front sensor (2) every one second.

```
/*
  Reads the SENSE front sensor and prints the value in the serial monitor
*/

#include "com202.h"

void setup()
{
  Serial.begin(9600);
  SerialBegin(1);
  Serial.println("Printing Front sensor values every one second");
}

int val = 0;

void loop()
{
    // sense front sensor number is 2
    val = StringToInt(GetInput("sense",1,2));
    Serial.print("Front sensor = ");
    Serial.println(val);
    delay(1000);
}
```

In this program, we use the two USARTs.

The instruction:

Serial.begin(9600);

initializes the USART0 for 9600 baud for communication with the PC.

The instruction:

SerialBegin(1);

initializes the USART1 for communication with the SENSE.

The instruction:

val = StringToInt(GetInput("sense",1,2));

gets the SENSE (ID=1) front sensor (sensor no. 2) value as a string, converts it to an integer and puts it into the variable **val**.

The program prints **"Front sensor ="** and the read value, and waits for one second.

1.3.3 Moving the SENSE to a wall

The following program moves the SENSE forward and stops when the SENSE is close to a wall.

In this program, we use the constant **STOP** with the stopping value of the front sensor.

The program moves the SENSE forward, enters into a loop until the front sensor value (val) is above STOP, stops the SENSE and exit.

```
/*
  Moves the SENSE forward and stops in front of a wall
*/
#include "com202.h"
#define STOP 350
int val = 0;

void setup() {
  SerialBegin(1);
}

void loop() {
    // to a wall and stop
  SetSpeed("sense",1,0,150); // set sense speed
  SetMotor("sense",1,0,"cw"); // sense forward

  while(val < STOP) {
    val = StringToInt(GetInput("sense",1,2)); // update front sensor val
  }

  SetMotor("sense",1,0,"off"); // sense stop
  exit(0);
}
```

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module into the right socket of the SENSE.

4. Enter the **Printing_Sense_Front_Sensor_Value** library in the **ARD-202** main library.

5. Double click on the **Printing_Sense_Front_Sensor_Value**.

6. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

7. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

8. Choose **Add File...** from the **Sketch** menu.

9. Browse to the Com202 library and select the **com202.h** file.

1.3.4 Printing front sensor value

10. Observe the following screen:

11. Scroll down to view the entire program.

12. Make sure that you understand all the program instructions.

13. Choose **Serial monitor** from the **Tools** menu.

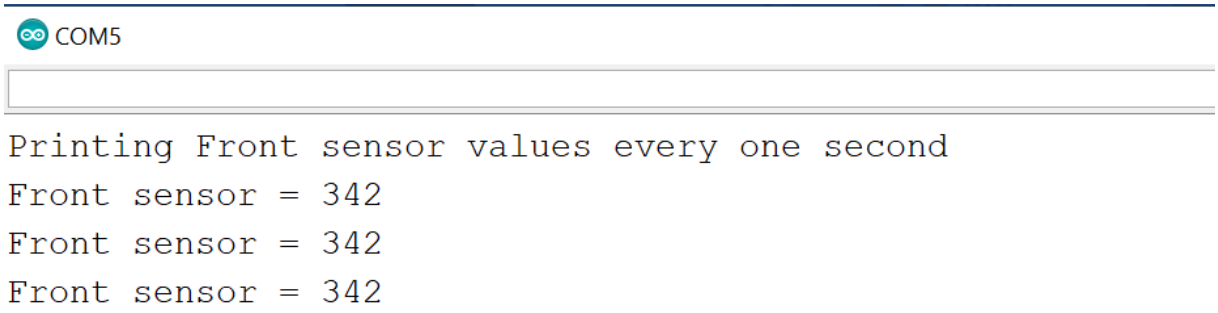
    The serial monitor screen will open.

14. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    A few seconds after the upload finishes, the following message with the front sensor value reading every one second should appear on the serial monitor:



    If it does not, press the **RST** (Reset) button on the Arduino board.

15. Put the SENSE on the table and put a wide object 30 cm in front of it.

    The front sensor values will be printed on the screen every one second.

16. Move the 'wall' object close to the SENSE and observe the displayed values.

17. Record the front sensor value in a distance of about 8 cm from the wall.

1.3.5 SENSE to a wall and stop

1. Enter the **Sense_to_a_wall_and_stop** library in the ARD-202 main library.

2. Double click on the **Sense_to_a_wall_and_stop**.

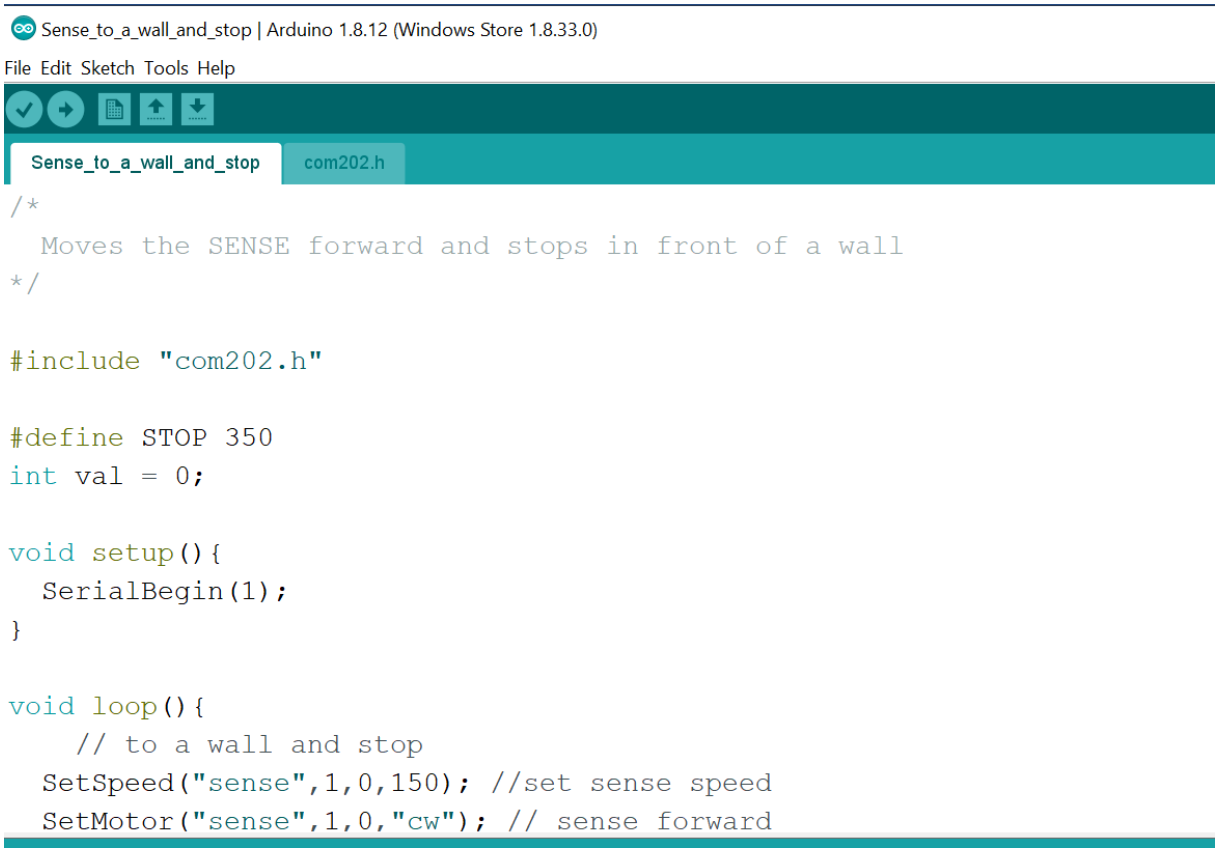
3. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

4. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

5. Choose **Add File...** from the **Sketch** menu.

6. Browse to Com202 library and select the **com202.h** file.

7. Observe the following screen:



8. Scroll down to view the entire program.

9. Make sure that you understand all the program instructions.

10. Put the SENSE on the table and put a wide object 30 cm in front of it.

11. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    The SENSE will move forward and stop about 8 cm from the wall.

12. Change the program so that the SENSE stops 5 cm in front of the wall.

    Check the SENSE movement.

1.3.6 SENSE to a wall and back

1. Change the program so the SENSE stops 5 cm in front of the wall, waits two seconds, moves backward for two seconds and then stops.

   Check the SENSE movement.

1.3.7 Endless loop

Most of the control and robotic programs are run in endless loop.

1. Delete the instruction **Exit(0);**

   The program will run in endless loop.

   Check the SENSE movement.

2. Run the SENSE on the floor without the communication cable.

1.3.8 Challenge exercise – Moving in a range of distance

Task 1: Improve the program so that the SENSE will move very slowly between 5 cm from the wall and 10 cm from the wall.

Experiment 1.4 – Movement Along a Black Line

Objectives:

- Program that reacts to sensors.
- Moving the SENSE to a black line.
- Moving the SENSE between lines.
- Moving the SENSE along a black line.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 battery module

Discussion:

In this experiment, we will move the SENSE to a black line and between two black lines. The position of the lines limits its motion. The SENSE changes direction when it finds a black line. This is an example of a system called a Manipulator.

We will learn how to read and react to the Bottom Line sensor.

1.4.1 Printing the value of the SENSE bottom sensor

The following program prints on the terminal screen the read values from the bottom sensor (1) every one second.

```
/*
  Reads the SENSE bottom sensor and prints the value in the serial monitor
*/
#include "com202.h"

void setup()
{
  Serial.begin(9600);
  SerialBegin(1);
  Serial.println("Printing Bottom sensor values every one second");
}
int val = 0;

void loop()
{
    // sense bottom sensor number is 1
    val = StringToInt(GetInput("sense",1,1));
    Serial.print("Bottom sensor = ");
    Serial.println(val);
    delay(1000);
}
```

In this program, we use the two USARTs.

The instruction:

Serial.begin(9600);

initializes USART0 for 9600 baud for communication with the PC.

The instruction:

SerialBegin(1);

initializes USART1 for communication with the SENSE.

The instruction:

val = StringToInt(GetInput("sense",1,1));

gets the SENSE (ID=1) bottom sensor (sensor no. 1) value as a string, converts it to an integer and puts it into the variable **val**.

The program prints the **"Bottom sensor ="** read value and waits for one second.

1.4.2 Moving the SENSE to a black line

The following program moves the SENSE forward and stops when the SENSE is on a black line.

In this program, we use the constant **STOP** with the stopping value of the bottom sensor.

The program moves the SENSE forward, enters into a loop until the bottom sensor value (val) is below STOP, stops the SENSE and then exits.

```
/*
  Moves the SENSE forward and stops on a black line
*/
#include "com202.h"
#define STOP 250
int val = 1000;

void setup() {
  SerialBegin(1);
}

void loop() {
    // to a black line and stop
  SetSpeed("sense",1,0,150); // set sense speed
  SetMotor("sense",1,0,"cw"); // sense forward

  while(val > STOP) {
    val = StringToInt(GetInput("sense",1,1)); // update bottom sensor val
 }

  SetMotor("sense",1,0,"off"); // sense stop
  exit(0);
}
```

Before proceeding, print two black lines as follows:

1.4.3 Moving the SENSE along a black line

To move the SENSE along a black line we use the **turn** procedures of the SENSE.

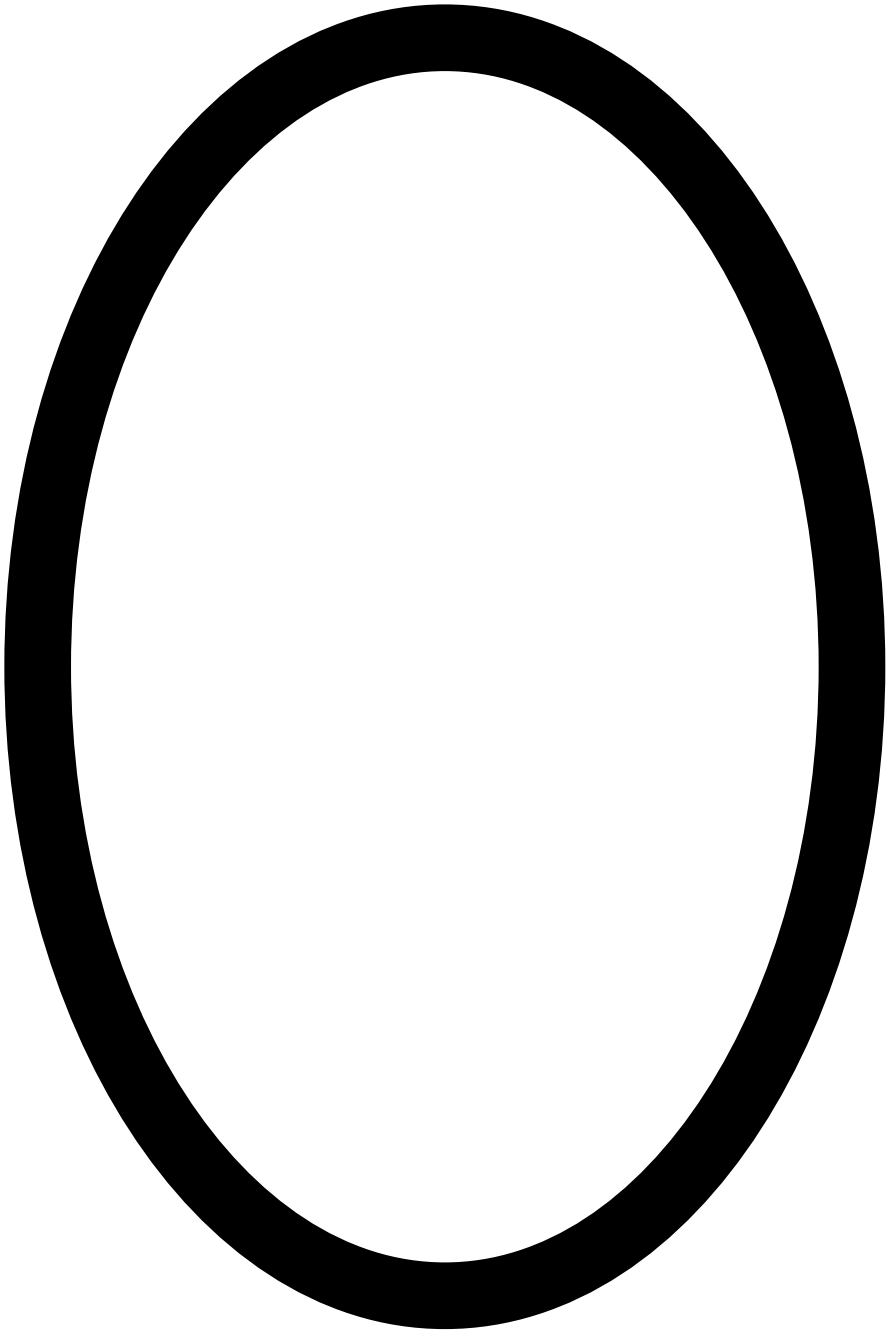
In turns, one wheel rotates and the other wheel stops. This way the SENSE still moves forward while turning.

In the main program, we do the movement according to the following idea:

Turning left until the SENSE find a black surface, and then turning right until the SENSE find a white surface.

```
/*
  Moves the SENSE along a black line
*/

#include "com202.h"

#define BLACK 250
int val = 1000;

void setup(){
  SerialBegin(1);
  SetSpeed("sense",1,0,150); // set sense speed
}

void loop(){
  SetMotor("sense",1,2,"cw"); // turn left
  SetMotor("sense",1,1,"off");

  while(val > BLACK){
    val = StringToInt(GetInput("sense",1,1)); // val = bottom sensor
   delay(100); //delay 100 ms
  }

  SetMotor("sense",1,1,"cw"); // turn right
  SetMotor("sense",1,2,"off");

  while(val <= BLACK){
   val = StringToInt(GetInput("sense",1,1)); // val = bottom sensor
   delay(100); //delay 100 ms
  }
}
```

Before proceeding, print a black line on a full page as follows:



The width of the line should be at least 3 cm.

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module into the right socket of the SENSE.

4. Enter the **Printing_Sense_Bottom_Sensor_Value** library in the **ARD-202** main library.

5. Double click on the **Printing_Sense_Bottom_Sensor_Value**.

6. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

7. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

8. Choose **Add File...** from the **Sketch** menu.

9. Browse to the Com202 library and select the **com202.h** file.

1.4.4 Printing bottom sensor value

10. Observe the following screen:

11. Scroll down to view the entire program.

12. Make sure that you understand all the program instructions.

13. Choose **Serial monitor** from the **Tools** menu.

    The serial monitor screen will open.

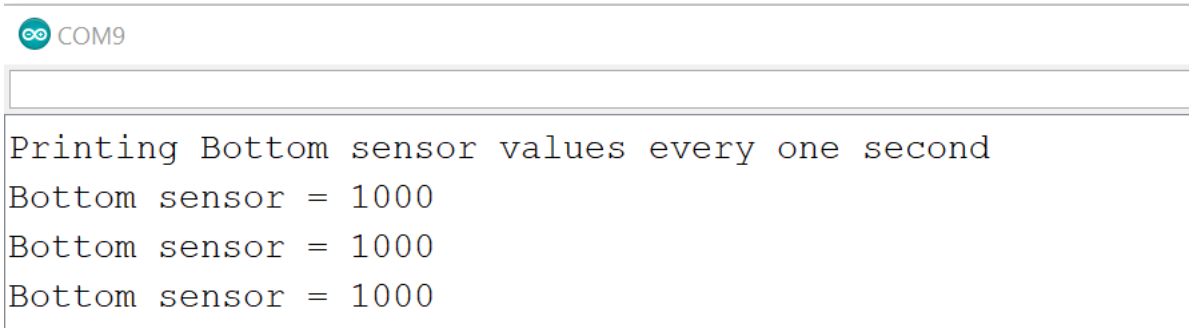
14. Put the SENSE on a white surface.

15. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    A few seconds after the upload finishes, the following message with the bottom sensor value reading every one second should appear on the serial monitor:



    If it does not, press the **RST** (Reset) button on the Arduino board.

16. Record the value of the sensor when the SENSE is on a white surface.

    The SENSE bottom sensor maximum value is 1000, which means white surface.

17. Put the SENSE on a black surface and observe the displayed values.

18. Record the value of the sensor when the SENSE is on a black surface.

1.4.5 Sense to a black line and stop

1. Enter the **Sense_to_a_black_line_and_stop** library in the ARD-202 main library.

2. Double click on the **Sense_to_a_ black_line _and_stop**.

3. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

4. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

5. Choose **Add File...** from the **Sketch** menu.

6. Browse to Com202 library and select the **com202.h** file.

7. Observe the following screen:

8. Scroll down to view the entire program.

9. Make sure that you understand all the program instructions.

10. Put the SENSE on a white surface with a black line in front.

11. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    The SENSE will move forward and stop on the black line.

1.4.6 SENSE to a wall and back

1. Change the program so that the SENSE stops on the black line, waits two seconds and then moves backward for two seconds.

   Check the SENSE movement.

1.4.7 Endless loop

1. Change the program so that the SENSE moves forward and stops when it meets the black line, moves back for 3 seconds, and moves forward again in an endless loop.

2. Delete the instruction **Exit(0);**

   The program will run in an endless loop.

   Check the SENSE movement.

3. Run the SENSE without the communication cable.

1.4.8 SENSE between two black lines

1. Change the program so that the SENSE moves forward and stops when it meets the black line, waits for 2 seconds, moves back until it meets the second black line, waits for 2 seconds, and moves forward again in an endless loop.

   Note:
   When the SENSE changes direction, it moves a little without checking its bottom sensor, to be sure that it is not on the black line.

   Add a short delay after each change direction instruction.

2. Put the SENSE on a white surface (with two black lines), between these two lines.

3. Click the **Upload** button.

   Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

   If the upload is successful, the message **"Done uploading"** will appear in the status bar.

   Check the SENSE movement.

4. Run the SENSE without the communication cable.

1.4.9 Challenge exercise – Between a wall and a black line

Task 1: Improve the program so that the SENSE will move between a wall in front and a black line at its back.

Note:
You have to use the Front sensor while moving forward. Pay attention to the compare signs (> or <).

1.4.10 SENSE along a black line

1. Enter the **Sense_along_a_black_line** library in the ARD-202 main library.

2. Double click on the **Sense_ along_a_black_line**.

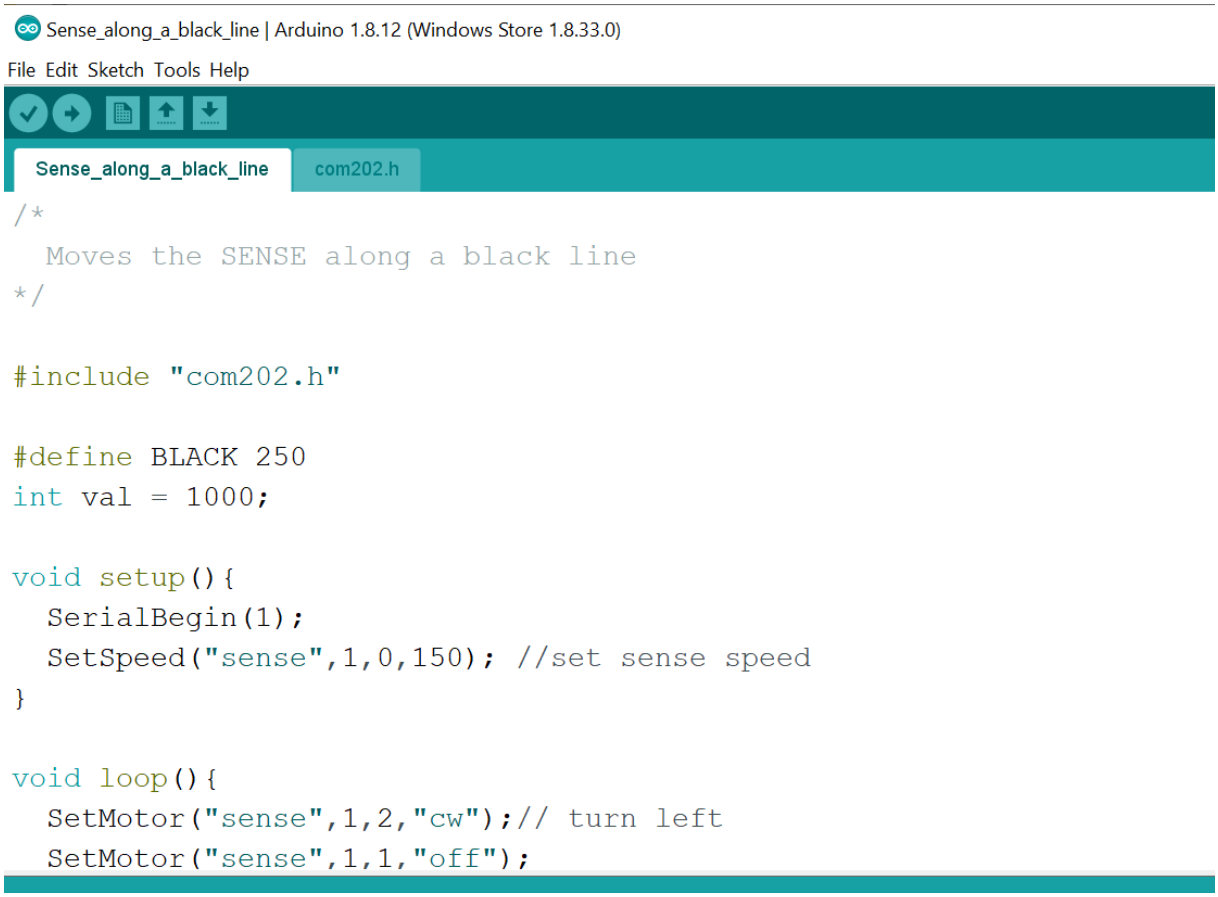
3. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

4. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

5. Choose **Add File...** from the **Sketch** menu.

6. Browse to Com202 library and select the **com202.h** file.

7. Observe the following screen:



8. Scroll down to view the entire program.

9. Make sure that you understand all the program instructions.

Note:
Pay attention to the compare signs (< and >).

10. Put the SENSE on a white surface near the black circle.

11. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    The SENSE should move along the black line.

12. Disconnect the communication cable from the ARD-202.

13. Change the value of the **BLACK** variable to create a smooth movement of the SENSE.

1.4.11 SENSE along a black line and stop

1. Enter the **Sense_along_a_black_line_and_stop** library in the ARD-202 main library.

2. Double click on the  **Sense_ along_a_black_line_and_stop**.

3. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

4. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

5. Choose **Add File...** from the **Sketch** menu.

6. Browse to Com202 library and select the **com202.h** file.

7. Observe the following screen:

8. Scroll down to view the entire program.

9. Make sure that you understand all the program instructions.

   In every cycle, the main procedure checks the distance from the wall and calls the **STOP** instruction when the SENSE is close to it.

   In control systems, we usually prefer that the OFF condition value will be different from the ON condition value. The reason is that we want to avoid having the system "bounce".

10. Put the SENSE on a white surface near the black circle.

11. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message "**Done uploading**" will appear in the status bar.

    The SENSE should move along the black line.

12. Disconnect the communication cable from the ARD-202.

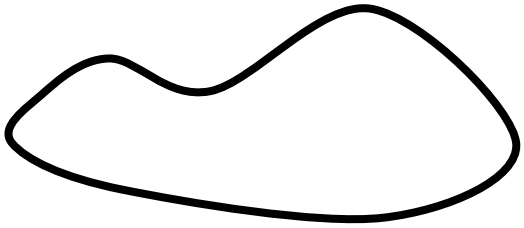
13. Put your hand in front of the SENSE, while it moves.

14. Change the values of the variables until the SENSE works well.

1.4.12 Challenge exercise – Along a complex black line

Task 1: Create different black lines for the SENSE and check its behavior. Improve the programs when needed.

The following is an example of a complex line:



Experiment 1.5 – Movement Along Walls

Objectives:

- Program that reacts to side sensors.
- Moving the SENSE along walls.
- Moving the SENSE along walls and stopping it.
- Moving the SENSE along walls and turning it around.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 battery module

Discussion:

In this experiment, we will move the SENSE along walls.

We will learn how to read and react to the Front right sensor.

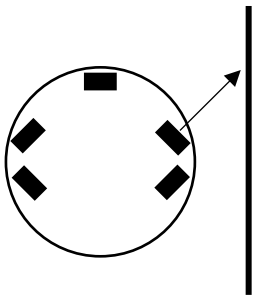
1.5.1 Movement along a wall

To move the SENSE along a wall, we use the same algorithm of moving the SENSE along a black line. We use the turn commands.

- Turn left when the SENSE is too close to the wall.
- Turn right when the SENSE is far from the wall.

To go along a wall on the right, we use the front side range sensor.

The side range sensors are installed in 45^{o} to the SENSE base.



When the SENSE turns to the right, the measured distance is smaller than when it turns to the left.

Think what will happen if the range sensor is parallel to the wall.

1.5.2 Printing the SNESE right front sensor value

The following program prints on the terminal screen the read values from the right front sensor (6) every one second.

```
/*
  Reads the SENSE right front sensor and prints the value in the serial monitor
*/

#include "com202.h"

void setup()
{
  Serial.begin(9600);
  SerialBegin(1);
  Serial.println("Printing right front sensor values every one second");
}

int val = 0;

void loop()
{
    // sense right front sensor number is 6
    val = StringToInt(GetInput("sense",1,6));
    Serial.print("Right front sensor = ");
    Serial.println(val);
    delay(1000);
}
```

In this program, we use the two USARTs.

The instruction:

Serial.begin(9600);

initializes USART0 for 9600 baud for communication with the PC.

The instruction:

SerialBegin(1);

initializes USART1 for communication with the SENSE.

The instruction:

val = StringToInt(GetInput("sense",1,6));

gets the SENSE (ID=1) front right sensor (sensor no. 6) value as a string, converts it to an integer and puts it into the variable **val**.

The program prints **"Front right sensor =**" read value and waits for one second.

1.5.3 Moving along walls

To move the SENSE along walls we use the turn commands of the SENSE.

In turns, one wheel rotates and the other wheel stops. This way the SENSE still moves forward while turning.

In the main program, we do the movement according to the following idea:

Turning right until the SENSE is close to the wall, and then turning left until the SENSE is far from the wall.

```
/*
  Moves the SENSE along walls on the right
*/

#include "com202.h"

#define WALL 300
int val = 0;

void setup(){
  SerialBegin(1);
  SetSpeed("sense",1,0,150); // set sense speed
}

void loop(){
  SetMotor("sense",1,2,"cw");// turn left
  SetMotor("sense",1,1,"off");

  while(val >= WALL){
   val = StringToInt(GetInput("sense",1,6)); // val = right front sensor
   delay(100); //delay 100 ms
  }

  SetMotor("sense",1,1,"cw");// turn right
  SetMotor("sense",1,2,"off");

  while(val < WALL){
   val = StringToInt(GetInput("sense",1,6)); // val = right front sensor
   delay(100); //delay 100 ms
  }
}
```

Note:
Pay attention to the compare signs (< and >).

Before proceeding, prepare a box for the SENSE to go around it.

Take care that the box is not black or with dark color. White box is better.

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module into the right socket of the SENSE.

4. Enter the **Printing_Sense_Bottom_Sensor_Value** library in the **ARD-202** main library.

5. Double click on the **Printing_Sense_Bottom_Sensor_Value**.

6. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

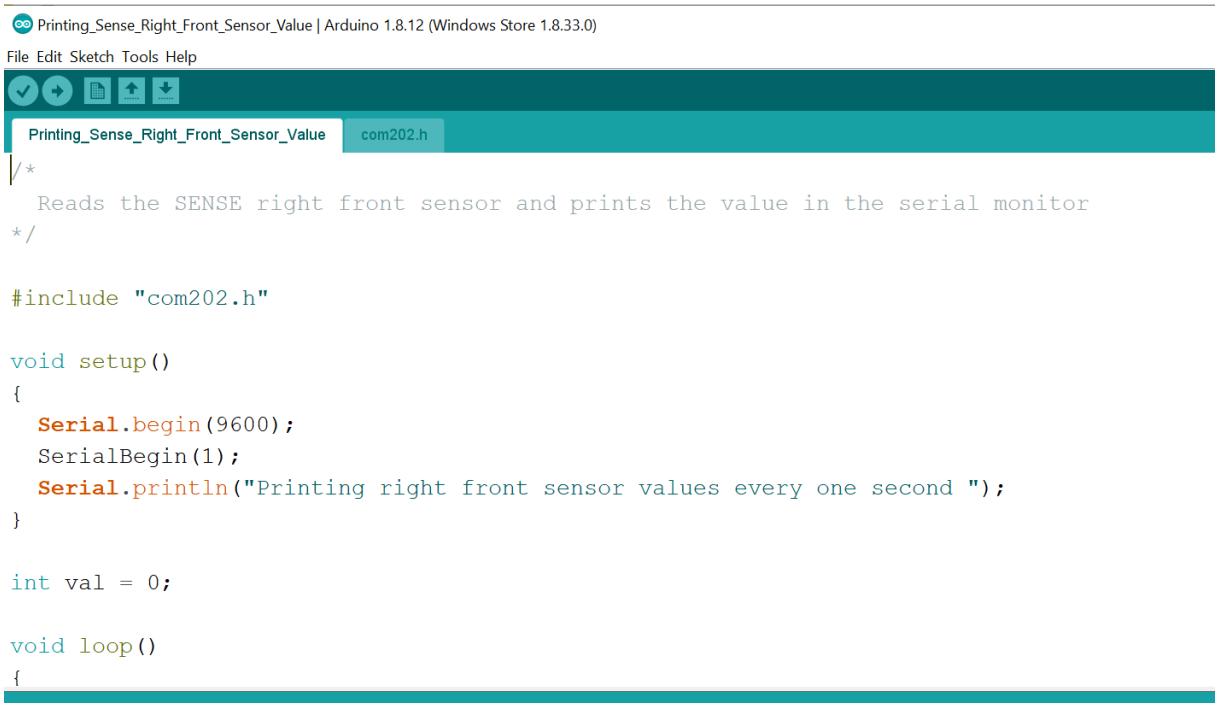
7. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

8. Choose **Add File...** from the **Sketch** menu.

9. Browse to the Com202 library and select the **com202.h** file.

1.5.4 Printing right front sensor value

10. Observe the following screen:



11. Scroll down to view the entire program.

12. Make sure that you understand all the program instructions.

13. Choose **Serial monitor** from the **Tools** menu.

    The serial monitor screen will open

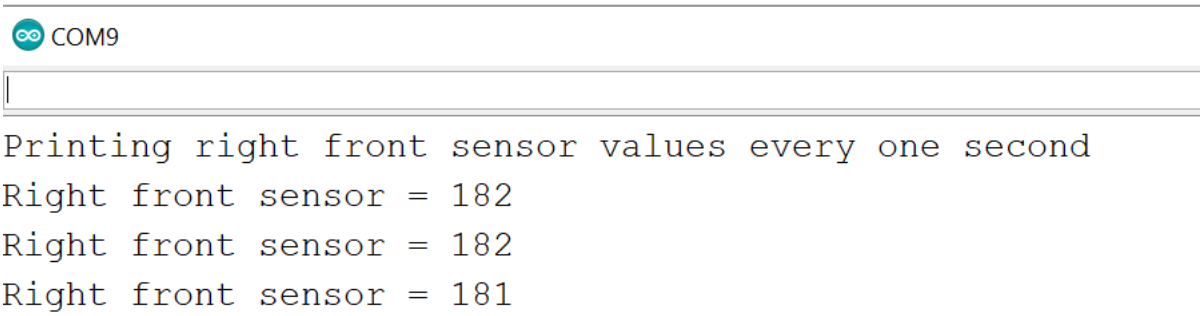
14. Put the SENSE on the left side of the box.

15. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    A few seconds after the upload finishes, the following message with the right front sensor value reading every one second should appear on the serial monitor:



    If it does not, press the **RST** (Reset) button on the Arduino board.

16. Record the value of the sensor when the SENSE is 4 cm parallel to the box.

1.5.5 SENSE along walls

1. Enter the **Sense_along_walls** library in the **ARD-202** main library.

2. Double click on the **Sense_along_walls**.

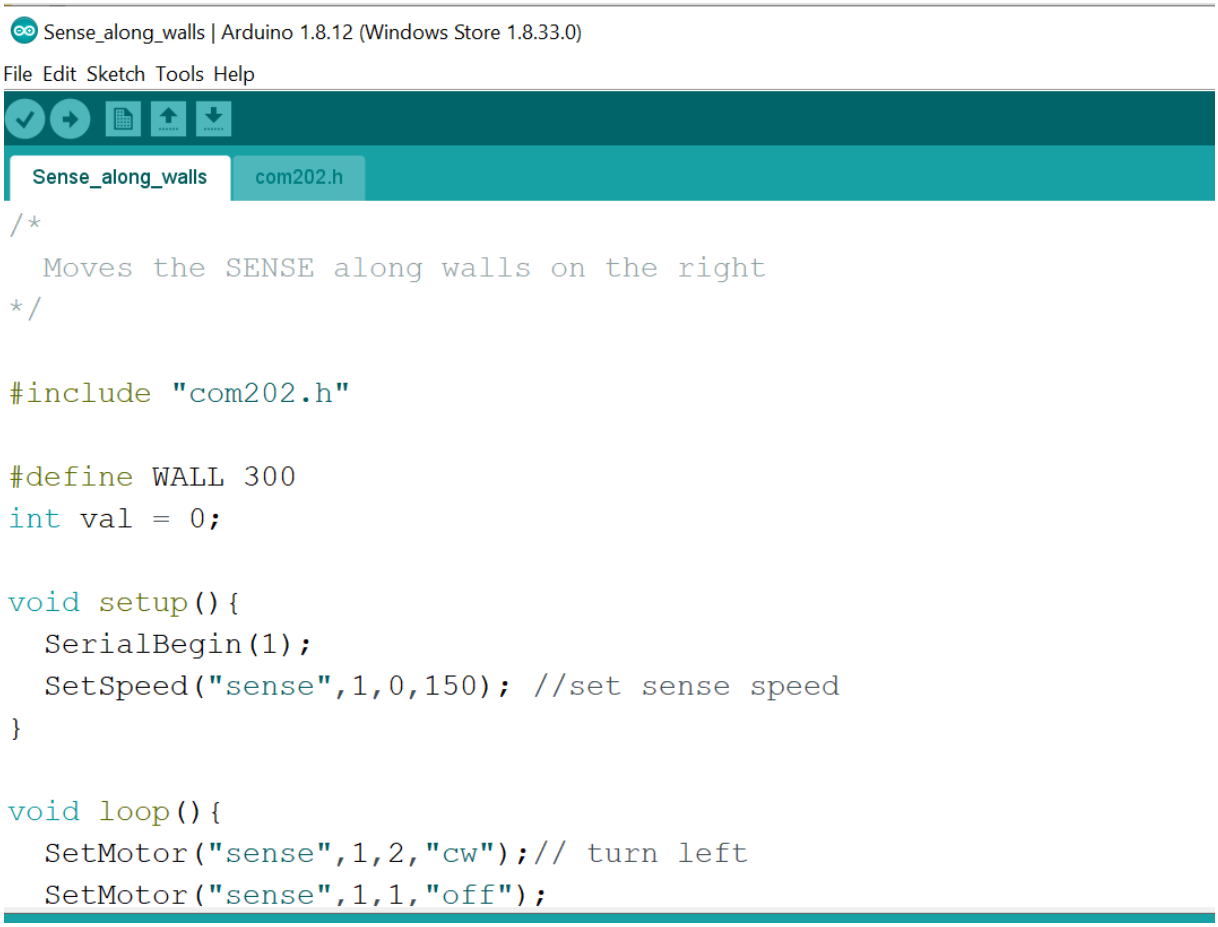
3. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

4. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

5. Choose **Add File...** from the **Sketch** menu.

6. Browse to Com202 library and select the **com202.h** file.

7. Observe the following screen:



8. Scroll down to view the entire program.

9. Make sure that you understand all the program instructions.

Note:
Pay attention to the compare signs (< and >).

10. Put the SENSE on the left side of the box.

11. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

12. The SENSE should move along the box walls.

13. Disconnect the communication cable from the ARD-202.

14. Change the value of the **WALL** variable to create a smooth movement of the SENSE.

1.5.6 SENSE along walls and stop

1. Enter the **Sense_along_walls_and_stop** library in the **ARD-202** main library.

2. Double click on the  **Sense_along_walls_and_stop**.

3. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

4. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

5. Choose **Add File...** from the **Sketch** menu.

6. Browse to Com202 library and select the **com202.h** file.

7. Observe the following screen:

8. Scroll down to view the entire program.

9. Make sure that you understand all the program instructions.

   In every cycle, the main procedure checks the distance from the wall and calls the **STOP** instruction when the SENSE is close to the wall.

   In control systems, we usually prefer that the OFF condition value will be different from the ON condition value. The reason is that we want to avoid having the system "bounce".

10. Put the SENSE on the left side of the box.

11. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    The SENSE should move along the box walls.

12. Disconnect the communication cable from the ARD-202.

13. Put your hand in front of the SENSE, while it moves.

14. Change the values of the variables until the SENSE works well.

1.5.7 Challenge exercises – Forward and along walls

Task 1: Improve the program so that the SENSE moves forward when it does not sense a wall on its right side.

The SENSE stops when it meets a wall, turns to the left and starts moving along this wall.

Challenge 1.6 – Counting

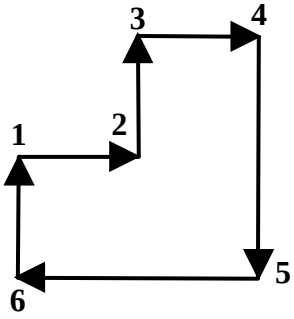
Draw block lines on a white paper as follows.

Create a program where the SENSE moves through the block lines and stops on the fourth line.

Use variables to count the lines.

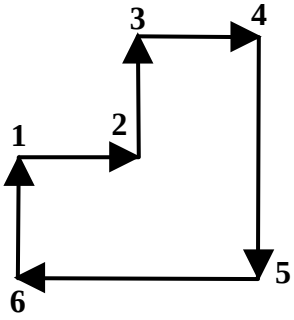
Challenge 1.7 – Automatic movement

Create a program where the SENSE moves according to the following figure:



Challenge 1.8 – Loops

Use loop commands to make the SENSE do the following cycles 3 times.



Challenge 1.9 – Loops and procedures

Convert each turn and forward movements into a procedure so that the main program will have only the loop and run procedure instructions.

The program should do the same as the program in challenge 1.8.

Challenge 1.10 – "Don't touch me" robot

Create a "Do not touch me" program.

The SENSE should move away when you bring your hand close to one of its range sensors.

Challenge 1.11 – Robots in a convoy

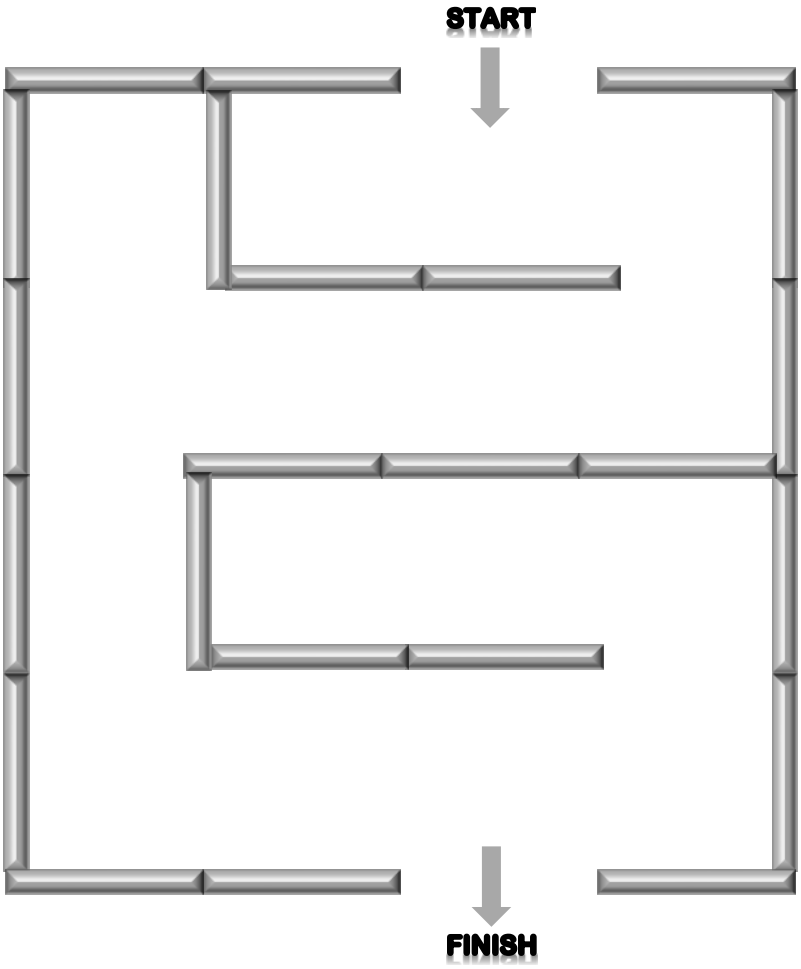
Put two SENSE robots on a black line.

The first SENSE should move along the black line and stop every 10 seconds.

The second SENSE should move along the black line and stop when it is close to the first SENSE.

Challenge 1.12 – Movement in a labyrinth

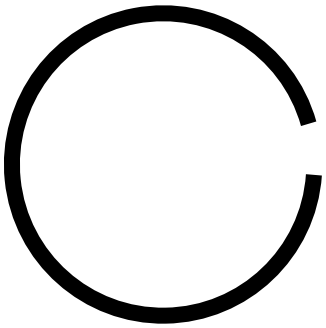
Build a labyrinth as follows:



Create a program where the SENSE moves from the START point to the FINISH point without touching the walls.

Challenge 1.13 – Exiting a circle

Draw a wide black line as follows:



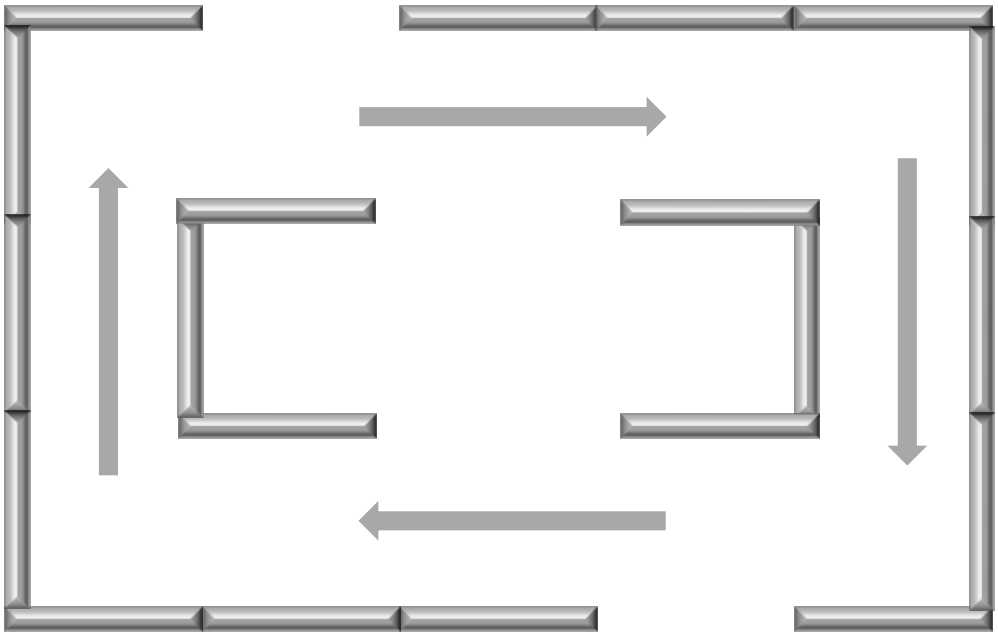
Put the SENSE inside the circle.

The SENSE should not cross the black line or move along the black line.

Create a program where the SENSE exits the circle according to the above rules.

Challenge 1.14 – Moving along corridors

Build the following corridor model:



The corridor's and the doors' widths are about 20 cm.

The SENSE should move in the corridor without getting out through the doors.

Create a program that answers this challenge.

Chapter 2 – Brain Units

2.1 Brain units

Some of the input units can have their own "brain". The NeuLog sensors are such brain units. They send to the control unit, upon request, processed data such as: temperature (^{o}C or ^{o}F), light intensity in Lux, distance in meters, etc.

The output units can also be brain units. For example, units that control the motor speed and direction, lamp intensity, servo motor angle, etc.

These brain units are connected in a chain to the main control unit, which communicates with them through messages.

Every brain unit has an ID number. Every message from the control unit starts with ID number. Only the brain unit with this ID number interprets the message and executes it.

This system construction is the way modern systems are built, and has important advantages:

1. It creates a system with much less wires. The wires go from one module to another and not from all modules to the control unit.

2. This kind of system can easily be changed and expanded, and does not depend on the control units number of inputs and outputs.

The experiments in this chapter use the following brain units:

- NeuLog light sensor (NUL-204)
- NeuLog sound sensor (NUL-212)
- NeuLog motion sensor (NUL-213)
- NeuLog magnetic sensor (NUL-214)
- Brain tracking unit (SNS-101)
- Brain gripper arm (SNS-167)

If you do not have them, you can read about them and move to chapter 3.

Chapter 3 experiments are with The SENSE robot and battery module.

2.2 NeuLog sensors as brain units

NeuLog sensors (Neuron Logger Sensors) are also brain units. Each sensor includes a tiny computer, which samples, processes and stores the sampled data. Each probe connected to the sensor is pre-calibrated in the factory and no further calibration is required.

The data provided by the sensor is processed digital data. The sensor includes different measurement ranges. Changing the measuring range or type of processing is done simply on the computer screen with NeuLog software.

The sensors are plugged to each other with almost no limitation on the composition and number of sensors in the chain.

NeuLog has over 50 different sensors. Some sensors perform as two to three sensors.

The SENSE has three sockets for NeuLog sensors.

Experiment 2.1 – Sound Sensor

Objectives:

- The sound sensor.
- Operating the SENSE by sound.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 Battery module
- NUL-212 NeuLog sound sensor

Discussion:

The sound sensor uses an internal microphone and special amplifier. Sound waves enter through the hole in the top of the sensor's plastic body so you should point that directly towards the sound source for best readings.

The sound sensor has two modes (ranges) of operation:

1. **Arbitrary analog units (Arb)** – An arbitrary unit indicates a number according to signal shape. At this mode, the sound is sampled and reconstructed as a signal.

2. **Decibel (dB)** – A unit of measure to show the intensity (loudness of sound). Please note that this is a logarithmic unit.

   At this mode, the wave is sampled and the average intensity (calculated by the sensor controller) is converted into dB value. 40 dB represents silence.

In this experiment, we shall use it at dB mode and we assume its ID is 1 as the default ID.

Selecting the range should be done by the NeuLog software.

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module into the right socket of the SENSE.

4. Plug the NUL-212 sound sensor into the front socket of the SENSE.

5. Enter the **Sense_to_a_wall_with_sound_sensor** library in the **ARD-202** main library.

6. Double click on the **Sense_to_a_wall_with_sound_sensor**.

7. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

8. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

9. Choose **Add File...** from the **Sketch** menu.

10. Browse to the Com202 library and select the **com202.h** file.

11. Observe the following screen:

12. Scroll down to view the entire program.

13. Make sure that you understand all the program instructions.

14. Put the SENSE on the table and put a wide object 30 cm in front of it.

15. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    The SENSE will not move.

16. Disconnect the SENSE from the computer.

17. Clap your hand or make a loud sound.

    The SENSE should move towards the wall and stop.

2.1.1 Challenge exercise – Wait for a sound

Task 1: Improve the program so:

(a) The SENSE will wait for a sound above 70 dB, then moves forward until it meets a wall and then stops for 5 seconds.
(b) It will wait again for the sound, moves backward until it reaches a black line and then stops for 5 seconds.
(c) Returns to the beginning.

Experiment 2.2 – Motion Sensor

Objectives:

- The motion sensor as distance sensor.
- Moving the robot according to the motion sensor.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 Battery module
- NUL-213 NeuLog motion sensor

Discussion:

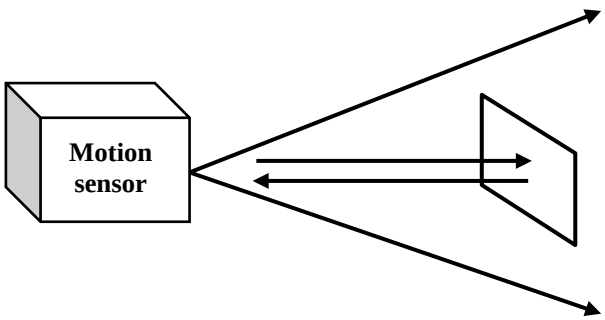
The motion sensor uses an ultrasonic transducer to both transmit an ultrasonic wave, and to measure its echo return. Objects in the range of 0.15 to 10 meters can accurately be measured to give distance, velocity, and acceleration readings using this method.

The motion sensor can collect data using the following measuring units:

- **Meters (m)** – The SI (International System of Units) distance unit
- **Meters/second (m/s)** – The SI velocity unit, which measures the distance traveled over time.
- **Meters/second^2 (m/s^2)** – The SI acceleration unit, which measures the change in velocity over time.

The motion sensor has two working ranges – one between 0.2 and 10.0 meters and one between 0.15 to 2 meters.

Ultrasonic waves are emitted from the sensor and spread out in a cone pattern at about 15° around the point of reference.



The ultrasonic transducer is a device that can convert pulse train to transmitted ultrasonic pulses. These pulses can sense and convert back to electronic pulse train by another similar ultrasonic transducer, or by itself.

The ultrasonic transducer is based on ceramic crystal, which is cut in a certain way and is placed between two metal plates. The crystal is characterized by the piezoelectric effect. Electrical field changes between the plates create mechanical vibrations in the crystal.

The crystal has a resonance frequency. The mechanical vibrations and electrical reactions depend on this resonance frequency.

Supplying pulses to the crystal of the ultrasonic transducer (in a rate according to its frequency) causes it to vibrate and to transmit these pulses as an acoustic sound. This sound cannot be heard because it is above the hearing frequency range (usually it is at 40KHz).

The acoustic sound can be converted back to electronic pulses by another ultrasonic transducer or by the transmitter when it stops transmitting. The acoustic pulses vibrate this transducer and these vibrations are turned into voltage pulses.

The speed of the ultrasonic wave is about 300 m/s because it is a sound wave.

For distance measurement, a burst of the transducer frequency wave is sent and the system measures the time between the sending and the receiving.

$S = 300 \cdot t$

Velocity is calculated by the difference between two successive distances divided by the time between the samples (according to the sampling rate).

Acceleration is calculated the difference between two successive velocities divided by the time between the samples (according to the sampling rate).

The motion sensor uses a very sophisticated method that enables it to measure long distance range with a low power of pulses.

In this experiment, we shall use it at distance range and we assume its ID is 1 as the default ID.

Selecting the range should be done with the NeuLog software.

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module above the motion sensor.

4. Plug the NUL-213 motion sensor into the right socket of the SENSE.

5. Enter the **Sense_to_a_wall_with_motion_sensor** library in the **ARD-202** main library.

6. Double click on the **Sense_to_a_wall_with_motion_sensor**.

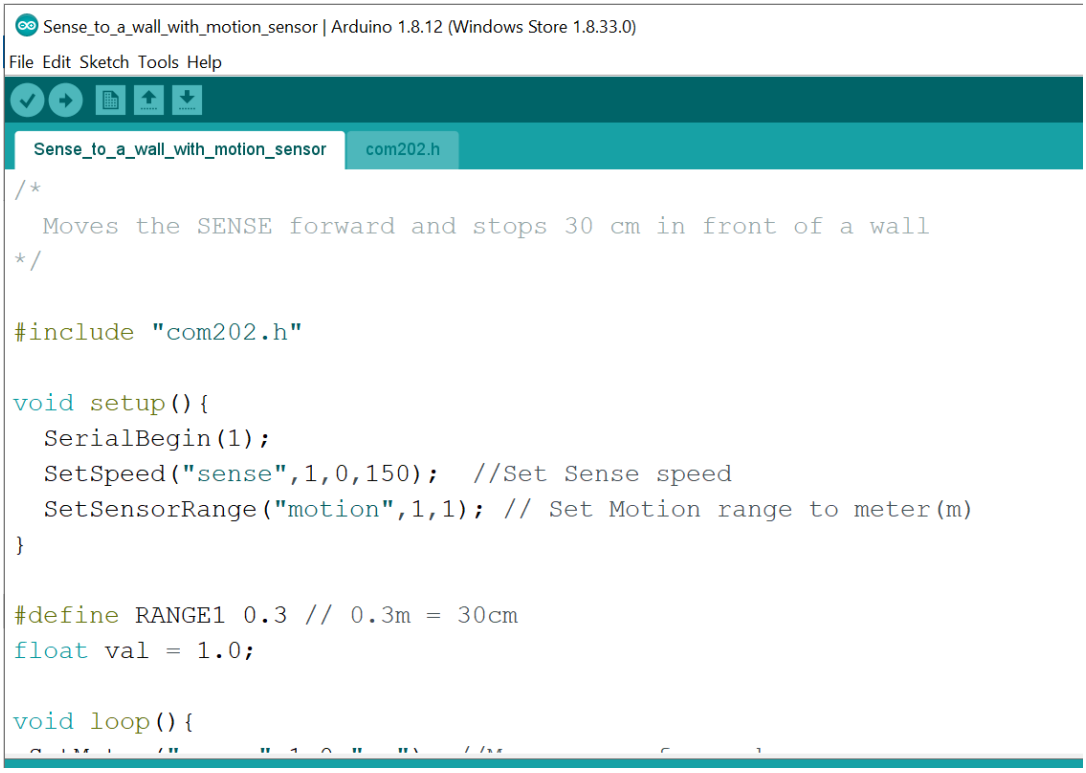
7. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

8. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

9. Choose **Add File...** from the **Sketch** menu.

10. Browse to the Com202 library and select the **com202.h** file.

11. Observe the following screen.



12. Scroll down to view the entire program.

13. Make sure that you understand all the program instructions.

14. Place the SENSE in front of a wall.

15. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

    The SENSE should move towards the wall and stop 30 cm away from it.

16. Disconnect the SENSE from the computer.

17. Operate the SENSE without the communication cable.

2.2.1 Challenge exercise – Moving in a distance range

Task 1: Improve the program so the SENSE will:

- move towards the wall,
- stop 30 cm in front of it,
- wait for 2 seconds,
- go backwards until a distance of 60 cm,
- wait for 2 second,
- return to the beginning and start again.

Experiment 2.3 – Brain Tracking Unit

Objectives:

- The brain tracking unit.
- Moving to an IR (infrared) transmitter.
- Following an IR transmitter.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 Battery module
- SNS-101 Brain tracking unit
- SNS-160 IR transmitter

Discussion:

2.3.1 IR Transmitter

The infra-red transmitter can be plugged into any of the SENSE sockets or in the backup battery socket to be followed by the brain tracking unit.

Infrared light is transmitted from a heat source. We cannot see the IR light. The frequency of this light is a little below the red light and this is why we call it infra (before) red.

The surrounding light does not affect this light much.

2.3.2 Brain tracking unit

The brain unit, in a rigid plastic case, can be plugged into one of the SENSE sockets.

The brain tracking unit has two IR (infrared) sensors that enables it to track the IR transmitter.

The two IR sensors are at the same line with an opaque partition between them.

When IR light falls on both of them, it means that the SENSE is in front of the IR light source.

When the SENSE is at angle to the light source, the IR light will fall only on one of the IR sensors.

The third IR sensor measures the environment IR light. The brain unit controller uses this measurement to eliminate the environment light.

The brain unit output is a binary number that describes the detection status of an IR transmitter. This number is converted to detection results as the following:

0 – None (00) – No IR transmitter light
1 – Right (01) – IR transmitter light on the right
2 – Left (10) – IR transmitter light on the left
3 – Front (11) – IR transmitter light at front

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module into the right socket of the SENSE.

4. Plug the SNS-101 Brain tracking unit into the front socket of the SENSE.

5. Enter the **Sense_tracking_IR_transmitter** library in the **ARD-202** main library.

6. Double click on the **Sense_tracking_IR_transmitter**.

7. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

8. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

9. Choose **Add File...** from the **Sketch** menu.

10. Browse to the Com202 library and select the **com202.h** file.

11. Observe the following screen:

12. Scroll down to view the entire program.

13. Make sure that you understand all the program instructions.

14. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

15. Disconnect the SENSE from the computer.

16. Place the SENSE on the floor.

    The SENSE is waiting for a signal from the IR transmitter.

17. Plug the IR transmitter into a backup battery.

18. Place the IR transmitter in front of the SENSE.

    The SENSE should stop rotating.

19. Move the IR transmitter to the left and to the right.

    The SENSE should follow it.

2.3.3 Challenge exercise – Tracking a robot with IR transmitter

Task 1: Improve the program to move the SENSE towards the IR transmitter. The SENSE waits when it does not detect the IR light.

Task 2: Improve the above program and procedures so the SENSE will stop in front of the IR transmitter.

Put the IR transmitter on a box or another SENSE that can be detected by the front sensor.

Experiment 2.4 – Brain Gripper Arm

Objectives:

- The brain gripper arm.
- Moving an object from one place to another.
- Drawing pictures with the brain gripper arm.

Equipment required:

- Computer
- SENSE autonomous
- ARD-202 coding unit
- BAT-202 Battery module
- SNS-167 Brain gripper arm
- A wooden rod
- A marker

Discussion:

2.4.1 Brain gripper arm

The brain gripper arm has two servo motors.

One servo motor moves the gripper up and down.

The second servo motor opens and closes the gripper.

A servo motor is a motor with feedback. The feedback can be voltage according to the motor speed or the shaft angle, electrical pulses according to the motor shaft rotation and direction, and more.

Each servo motor of the gripper arm has transmission gear and potentiometer. The potentiometer consists of a variable resistor that create variable voltage according to the servo motor shaft angle.

The brain controller of the gripper arm gets the required angle of the shaft. It turns the motor CW (Clock Wise) or CCW (Counter Clock Wise) until the potentiometer voltage suits this angle.

It checks the shaft angle all the time. If it changes mechanically, the controller will turn the motor ON to return the shaft to the right position.

Procedure:

1. Connect the ARD-202 to your computer using the USB communication cable. The power LED (labelled PWR) should turn on.

2. Check that the COM-202 is connected to the Arduino board (as described in section 1.8) and plug the COM-202 card into the left socket of the SENSE.

3. Plug the BAT-202 battery module into the right socket of the SENSE.

4. Plug the SNS-167 Brain gripper arm into the front socket of the SENSE.

5. Enter the **Sense_with_brain_arm** library in the **ARD-202** main library.

6. Double click on the **Sense_with_brain_arm**.

7. Choose **Board** from the **Tools** menu to select the board that corresponds with your Arduino board (see section 1.14 if necessary).

8. Choose **Port** from the **Tools** menu to select the serial device of the Arduino board. (see section 1.15 if necessary).

9. Choose **Add File...** from the **Sketch** menu.

10. Browse to the Com202 library and select the **com202.h** file.

11. Observe the following screen:

12. Scroll down to view the entire program.

13. Make sure that you understand all the program instructions.

    The program should do the following:

    (a) Opens the gripper
    (b) Raises the arm
    (c) Moves the SENSE forward and stops when the wooden rod is standing between the gripper fingers
    (d) Lowers the arm to mid position
    (e) Closes the gripper
    (f) Raises the arm
    (g) Moves forward for 2 seconds
    (h) Lowers the arm to mid position
    (i) Opens the gripper
    (j) Moves backward for 2 seconds

14. Click the **Upload** button.

    Wait a few seconds – you should see the RX and TX LEDs flashing on the board.

    If the upload is successful, the message **"Done uploading"** will appear in the status bar.

15. Disconnect the SENSE from the computer.

16. Check that the SENSE performs its mission.

17. Change the program to run in an endless loop.

2.4.2 Challenge exercises – The SENSE with a gripper arm

Task 1: Change the last program to use functions instead of chain of instructions. Put the delays in the functions.

Task 2: Change the program to make the SENSE to rotate in about 90^{o} with the raised wooden rod before moving forward with it.

Task 3: Plug Sound sensor to the SENSE and make it wait for hand clapping before picking up the wooden rod.

Task 4: Make the gripper hold a marker manually.

Place the SENSE on wide white paper attached to the ground or to the desk.

Build some drawing programs.

Chapter 3 – Autonomous Vehicle Challenges

3.1 Autonomous vehicles

We live in an era with autonomous vehicles, machine learning and artificial intelligence. This is a world where machines are making decisions based on software and programming; and this is just the beginning.

We can understand this world and the occurring changes by developing programs similar to those that operate autonomous vehicles.

The SENSE is a tool for such a challenge.

This chapter introduces several challenge autonomous exercises. The idea is to let the user think about algorithms and solutions for these challenges.

3.2 Programming tips

The challenge exercises in this chapter are built as flowcharts that help the user to solve the challenge exercise without guidance.

The flowchart is the map of the program.

We use variables in the flowcharts, called memories (Mem1, Mem2, etc.)

Name these variables as was done in chapter 1 and 2 programs.

The C language is rich and a powerful coding program.

It has many functions and options.

Try to work more with functions instead with long chains of instructions.

There are multiple solutions. Try to find the most efficient one.

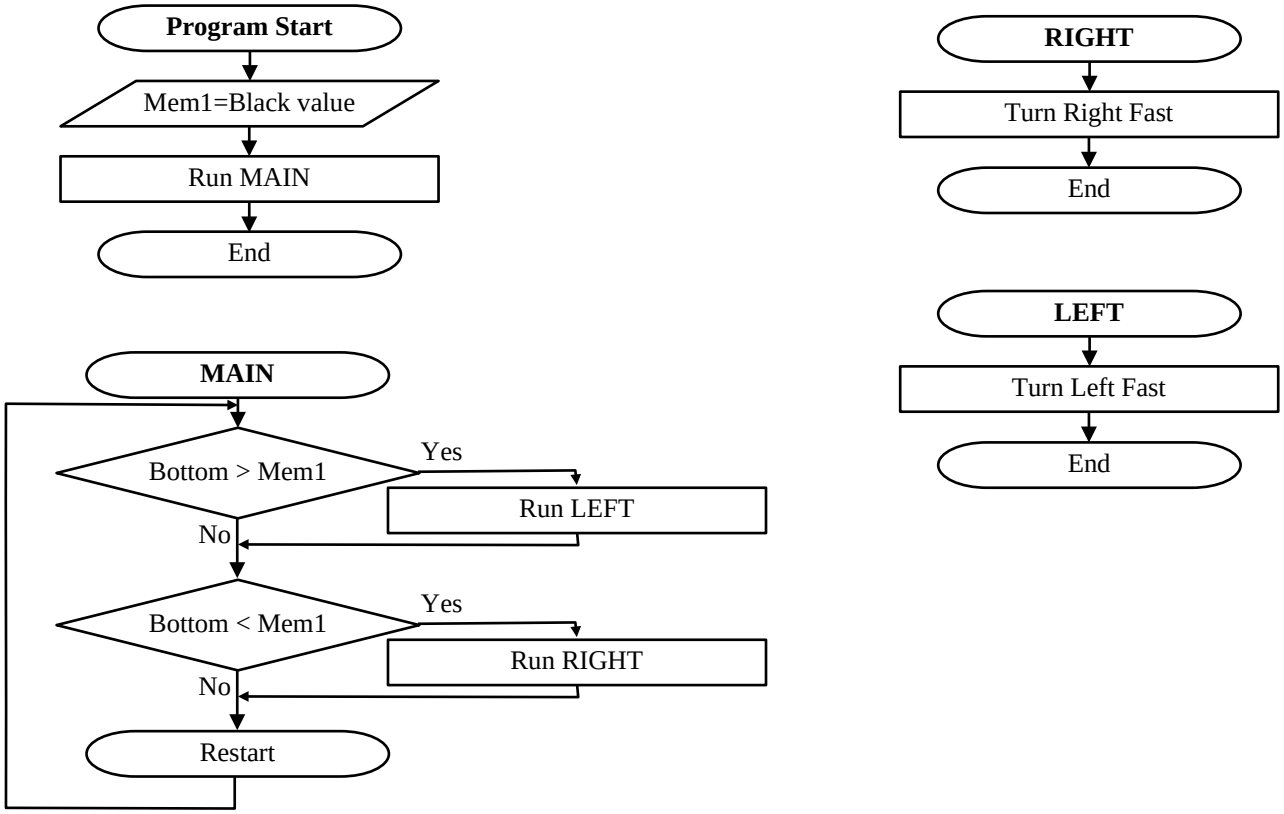
Do not be discouraged if you do not succeed on the first try. Keep on trying until you do.

Good Luck!!

Challenge 3.1 – Along black lines

3.1.1 Left and right along a black line

The following is a flowchart of a simple movement along a black line program.



The SENSE moves by swinging on the edge of the black line.

Place the SENSE on the black line, read the bottom sensor value and set it in the program. This value may be different from one SENSE to another.

Convert the flowchart to a C language program.

Download, run and check to SENSE movement.

3.1.2 Smooth movement along a black line

In order to get a smoother movement, we can replace one of the turn instructions with a deviate instruction. This depends on the SENSE movement direction.

When the SENSE moves counterclockwise, we shall replace the **Right turn** command with the **Right deviate** command.

When the SENSE moves clockwise, we shall replace the **Left turn** command with the **Left deviate** command.

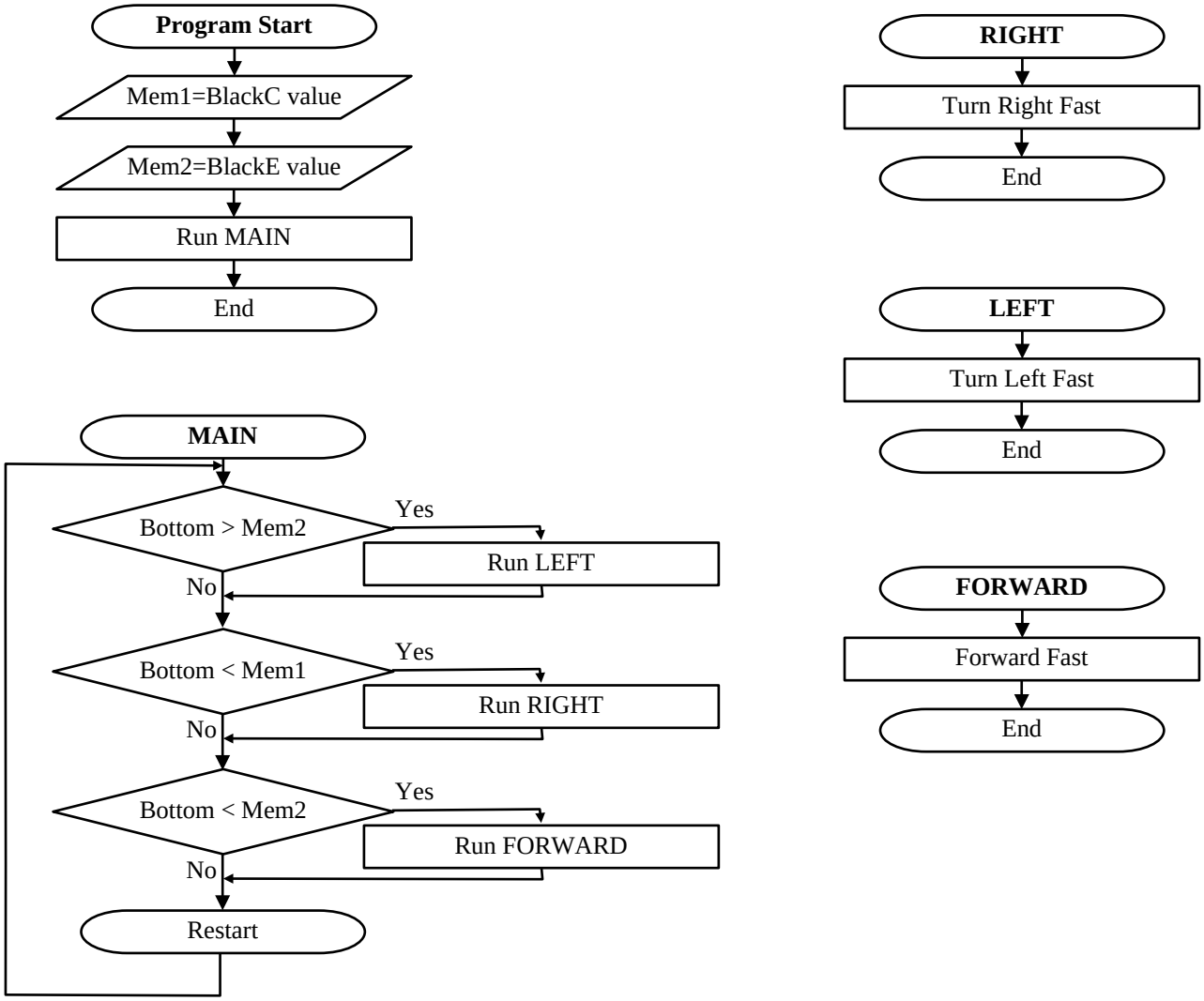
Change the program accordingly, download, run and check to SENSE movement.

3.1.3 Adding a Forward movement

In **Direct** mode, check the bottom sensor value when it is above the center of the black line and when it is closer to the edge of the line.

We shall call the read value at the center of the black line **BlackC** and the black value close to the edge **BlackE**.

The following program moves the SENSE forward when it is on the edge of the black line.



Analyze the flowchart.

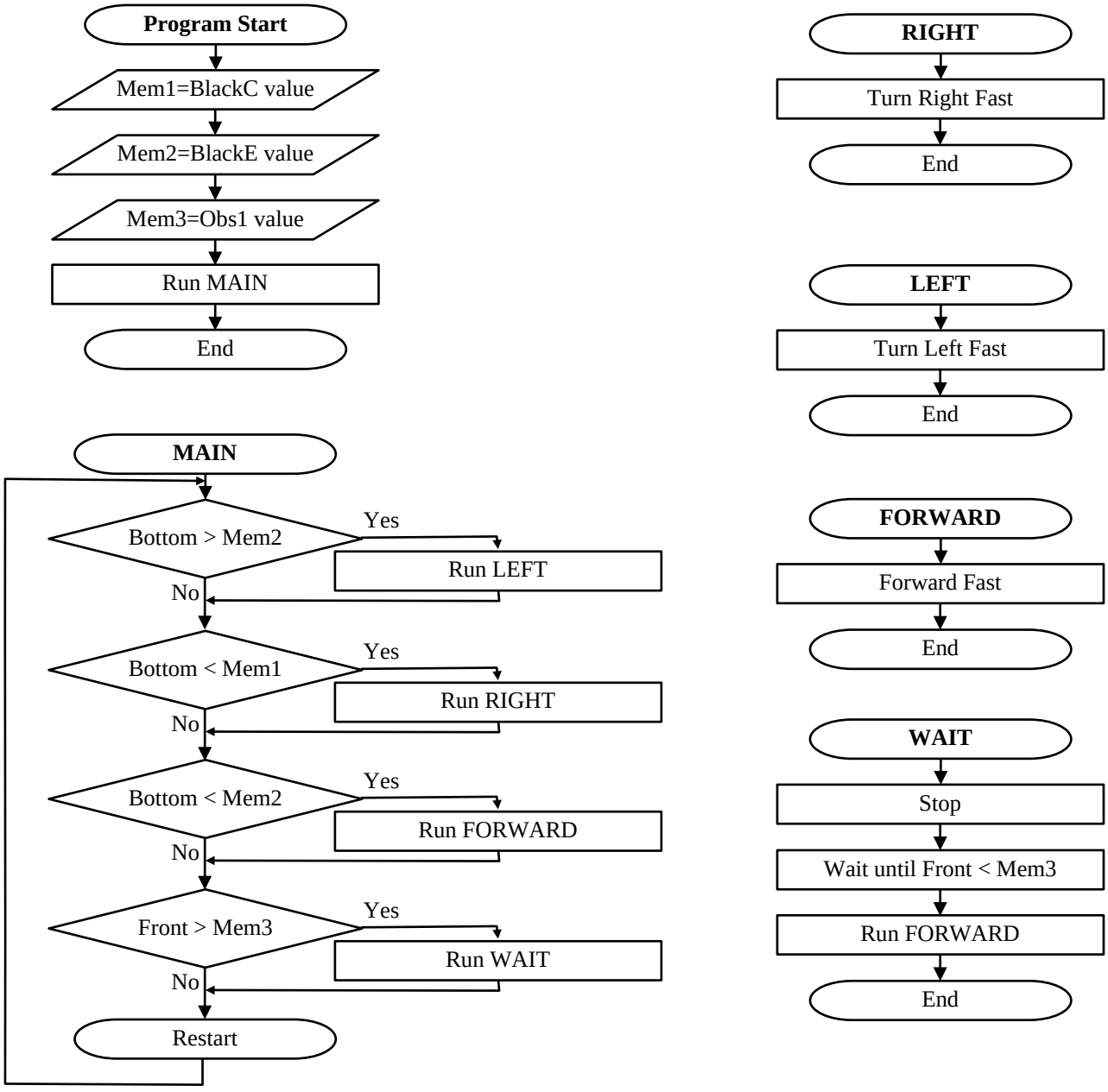
Change the program accordingly, download, run and check to SENSE movement.

3.1.4 Along a black line with a stop in front of an obstacle

Improve the previous program so the SENSE will stop in front of an obstacle until the obstacle is removed.

Put your hand in front of the SENSE and in **Direct** mode, check the front sensor value. We shall call the read value **Obs1**.

The following program moves the SENSE forward when it is on the edge of the black line.

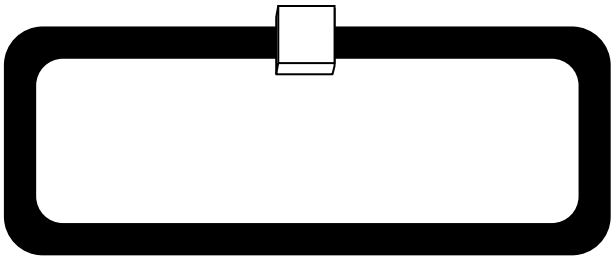


Analyze the flowchart. Change the program accordingly, download, run and check to SNESE movement.

Challenge 3.2 – AGV (Automatic Guided Vehicle)

An AGV is a vehicle or a cart that moves along guidelines. It is very popular in factories where row materials or sub-assembly systems need to be transported from one station to another.

Create the following line.



Put a small box on it as in the above picture.

Write a program that moves the SENSE along the line and stops in front of the box for 5 seconds, turns around, moves in the other direction and vice versa.

The SENSE moves on the outer edge.

For this task we have two movements – clockwise and counterclockwise.

The main program should know what the current movement is. To determine that, we use what we call a flag. The main program operates the required procedure according to the value of a certain variable.

The value of this variable is changed when changing direction is needed.

Analyze the following flowchart. Memory 4 is the flag variable.

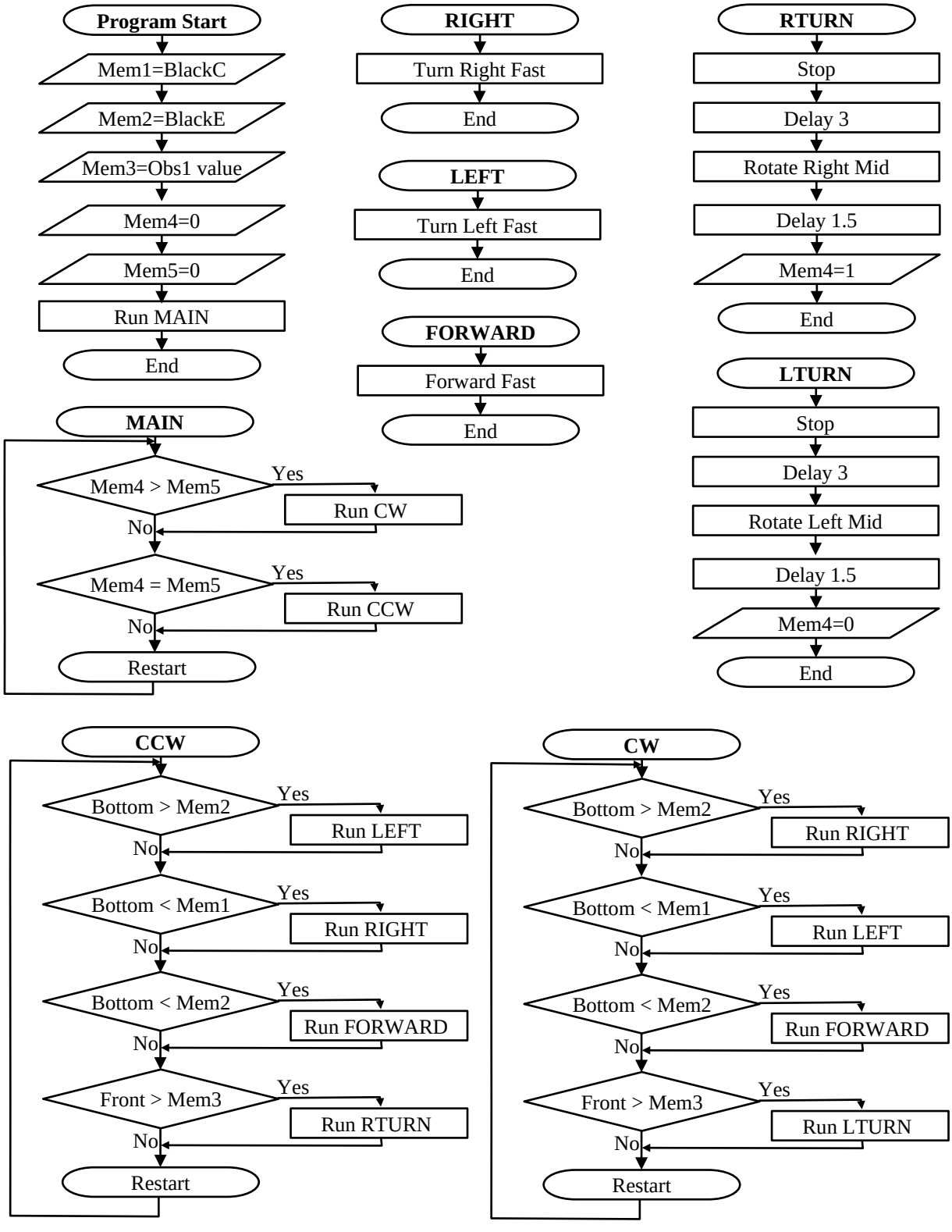
Build the program accordingly, download, run and check to SENSE movement.

Every SENSE behaves a little differently.

Adapt the program to your SENSE's sensors and behavior.

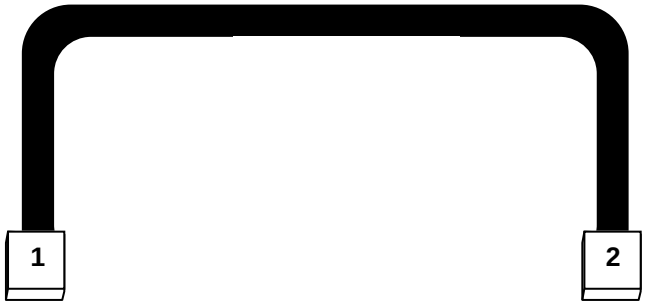
Take care to stop in front of the box in a distance that enables the SENSE to rotate.

AGV program flowchart



Challenge 3.3 – AGV between stations

Create the following line with the boxes.



Write a program where the SENSE moves from one station to another along the lines in this order: 1-2-1-2-…

The SENSE stops at each station and moves to the next station when you put your hand close to the right back sensor.

Hint:

Use the previous AGV program for SENSE movement.

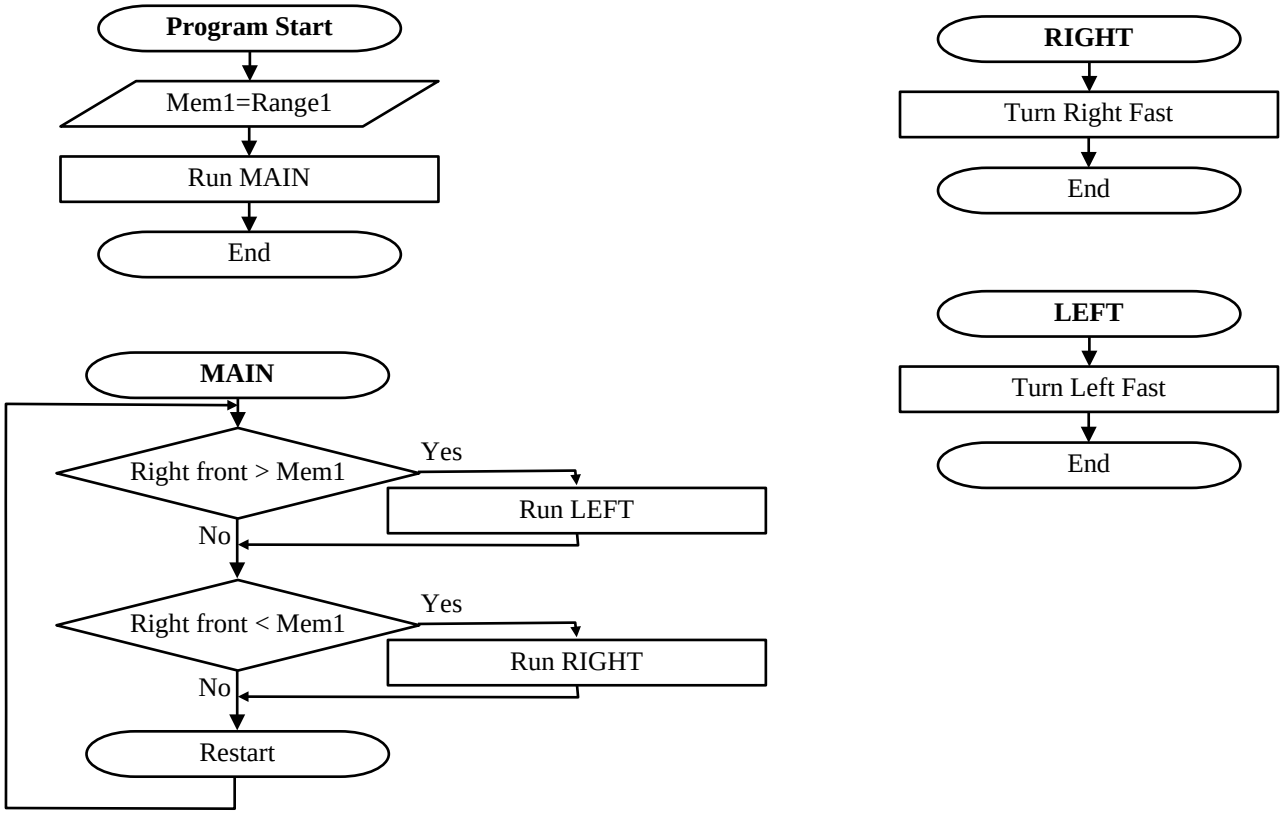
Challenge 3.4 – Along a building block

The following exercises deals with methods of moving along walls and around a building block.

Use at least 40 X 40 cm box as a simulation of a building block.

3.4.1 Left and right along walls

The following is a flowchart of a simple movement along walls program.



The SENSE moves by swinging along a wall on its right side.

Place the SENSE near the box on its right side, read the right front sensor value and set it in the program. This value may be different from one SENSE to another.

Download, run and check the SENSE movement.

3.4.2 Smooth movement along a black line

In order to get a smoother movement, we can replace one of the turn instructions with a deviate instruction. This depends on the SENSE movement direction.

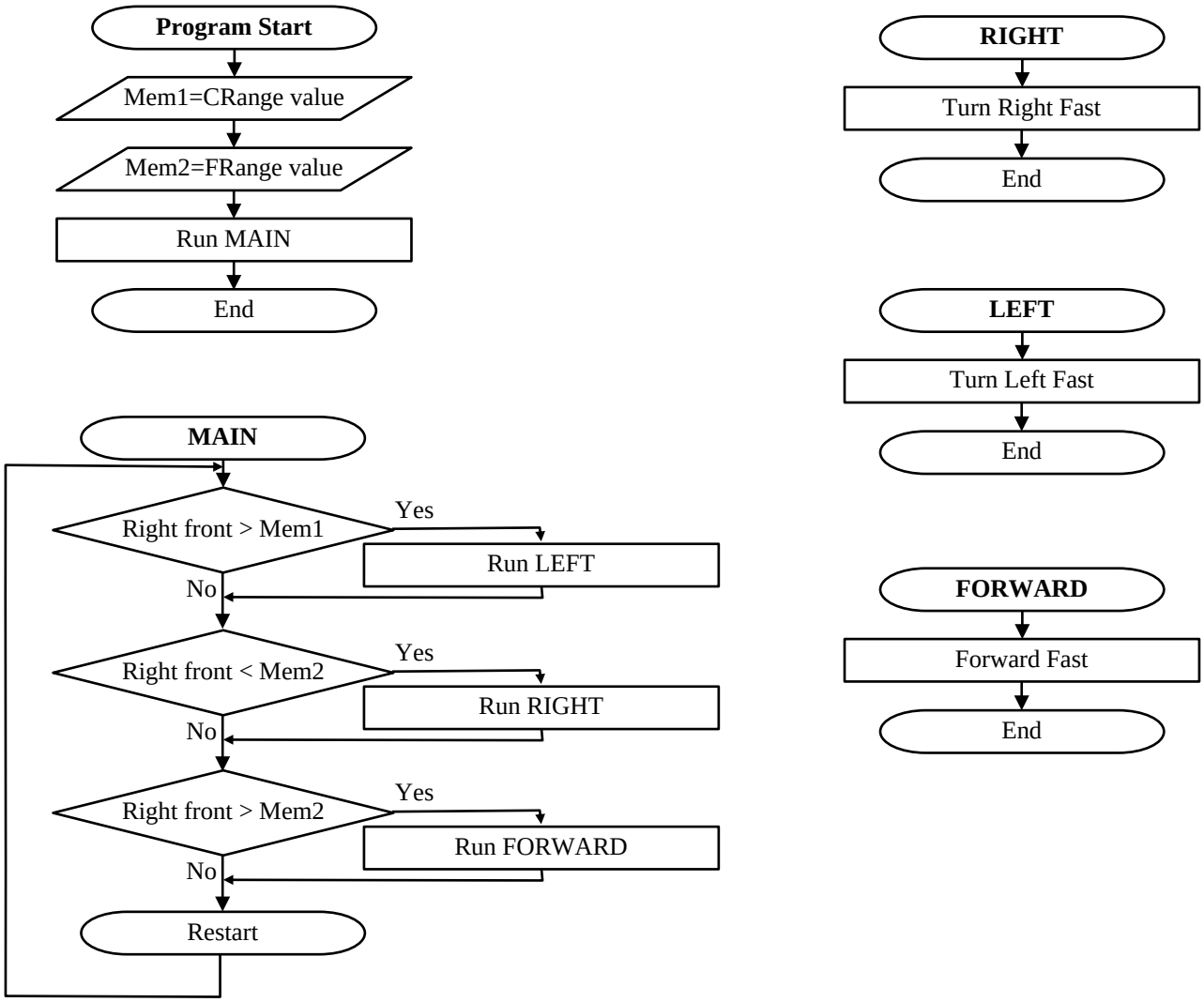
With the above program, the SENSE moves clockwise, so we shall replace the **Left turn** command with the **Left deviate** command.

Change the program accordingly, download, run and check the SENSE movement.

3.4.3 Adding a forward movement

We can use two range values – **Crange** (close range) and **FRange** (far range).

The following program moves the SENSE forward when it is between CRange and FRange.



Analyze the flowchart.

We have to remember that the right front value increase when the SENSE is closer to the wall.

Determine the **CRange** value as the **Range1** value of the previous program.

Determine the **Frange** value as **CRange** – 10.

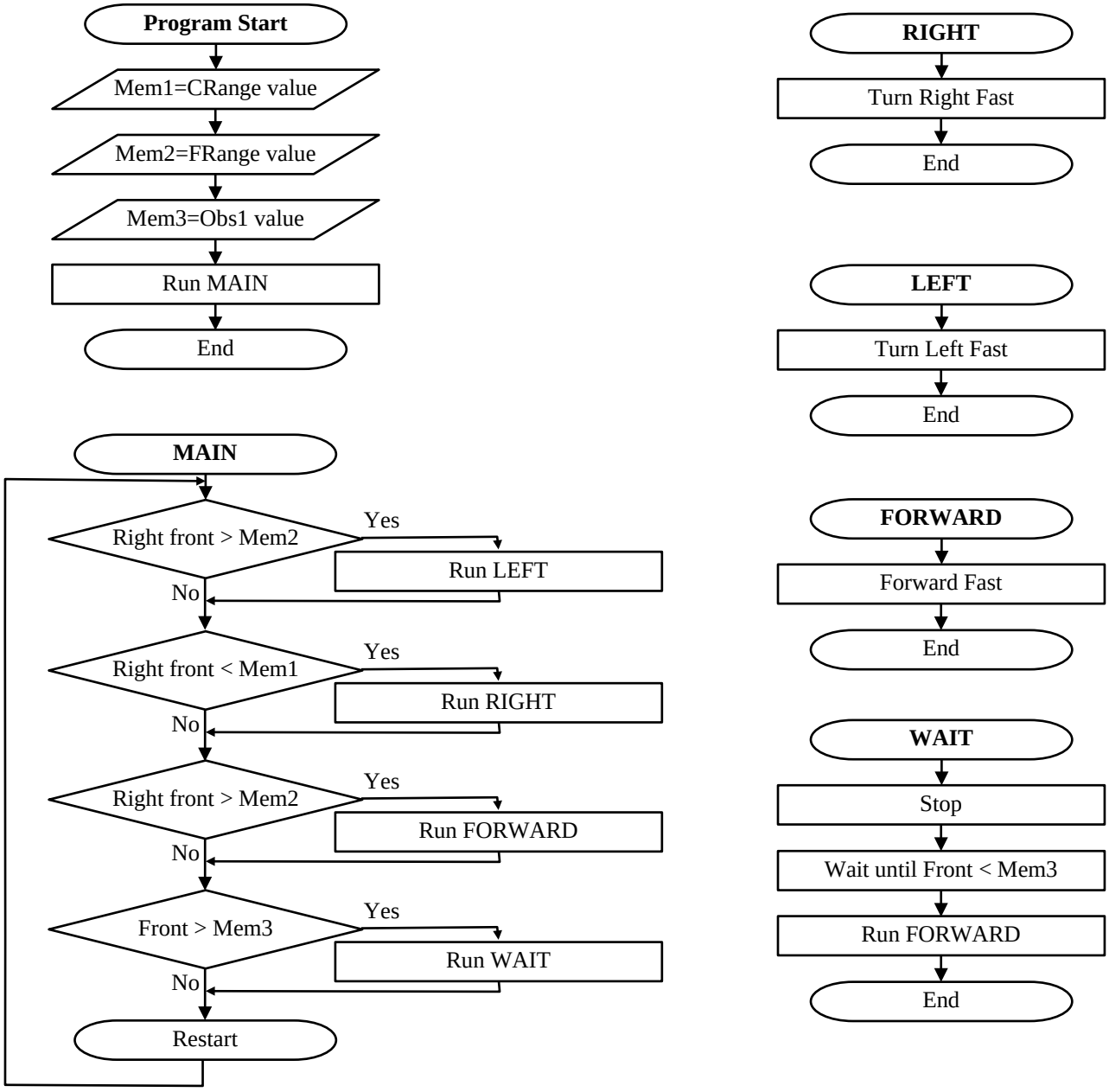
Change the program accordingly, download, run and check the SENSE movement.

3.4.4 Along a wall with a stop in front of an obstacle

Improve the previous program to stop in front of an obstacle until the obstacle is removed.

Put a small box in front of the SENSE and check the front sensor value in **Direct** mode. We shall call the read value **Obs1**.

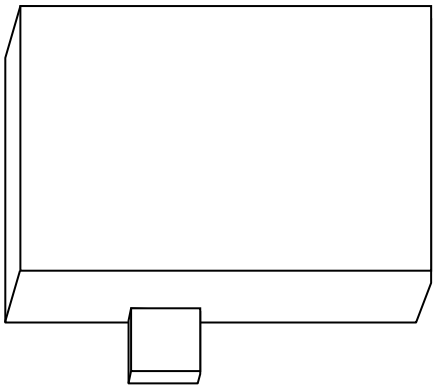
The following program moves the SENSE forward when it is on the edge part of the black line.



Analyze the flowchart. Change the program accordingly, download, run and check the SENSE movement.

Challenge 3.5 – Along a building block and bypass cars

Put an obstacle (simulates a car), as described in the following picture.



Write a program, as in challenge 3.4.4, where the SENSE bypasses the obstacle. The SENSE should return to the right only after passing the obstacle.

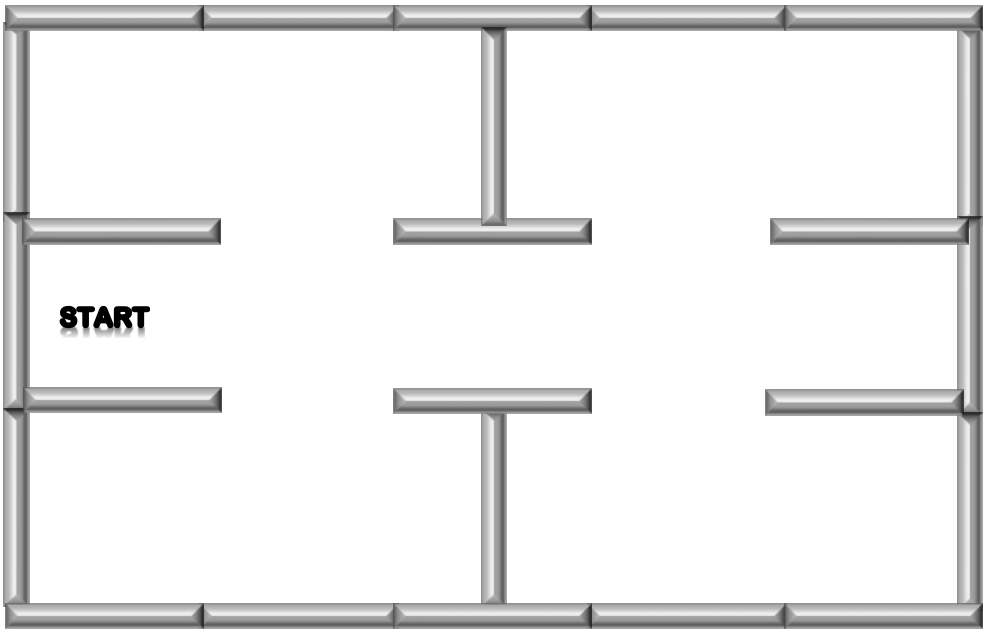
Hint:

Replace the **WAIT** procedure with the **TURN** procedure.

The **TURN** procedure rotate to the left until the front sensor is below the **Obs1** value.

Challenge 3.6 – Autonomous museum guard

Build a model of a museum with rooms and corridors as follows:



Create a program where the SENSE moves along the walls through the museum rooms. The starting point is at the START position.

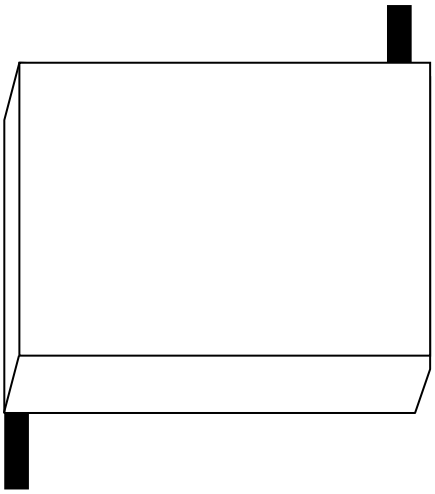
Hint:

The program of challenge 3.5 can serve as a solution for this task.

You have to adapt the memory values to the model.

Challenge 3.7 – Along a building block with stop sign

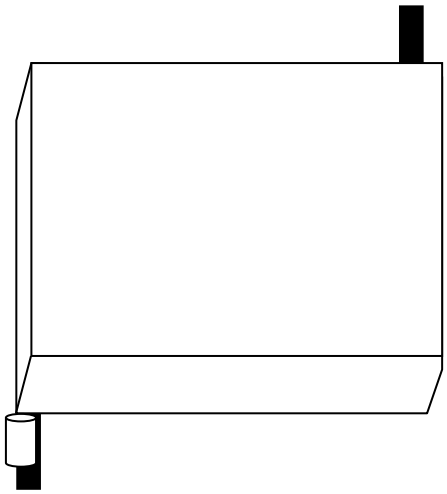
Put black lines (simulates stop signs) at the corners, as described in the following picture.



Write a program, as in challenge 3.4, where the SENSE stops for 3 seconds when it reaches the black lines.

Challenge 3.8 – Along a building block with pedestrian crossing

Put a rod (simulates a pedestrian crossing) at one of the corners, as described in the following picture.



Write a program, as in challenge 3.4, where the SENSE stops for 3 seconds it reaches the black line.

The SENSE does not move on if an obstacle is in front of it.

Challenge 3.9 – Guarding a building block

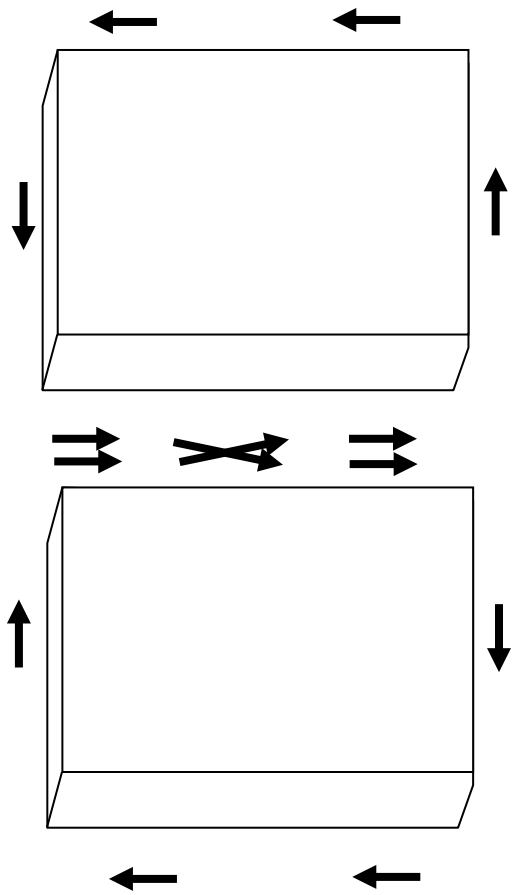
Write a program, as in challenge 3.4, where the SENSE stops for 10 seconds after each round.

The program has to count the corner turns.

Hints:

- The SENSE has two range sensors on each side.
- The movement along the wall is according to the front side range sensor.
- When the SENSE reaches a corner, the back-side sensor moves away from the wall.
- The program should check the value of the back-side sensor.
- When the value is low (the sensor is far from the wall), the SENSE should call a turn procedure that increases the corner number.
- After counting four corners, the SENSE should stop for 10 seconds and then starts again.

Challenge 3.10 – Guarding two buildings

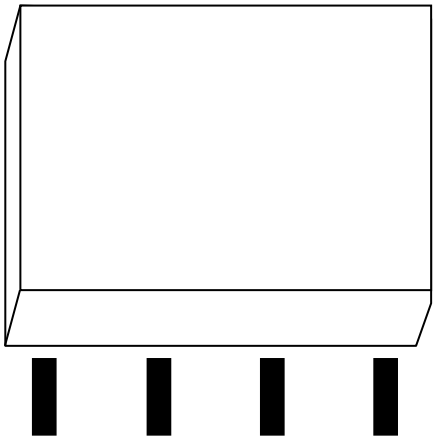


Write a program where the SENSE moves around two buildings as in the above picture. The SENSE starts at the road between the two blocks.

The program has to count the corner turns and to change from moving counterclockwise around one building to moving clockwise around the other building.

Challenge 3.11 – Taxi driver

Put black lines along the building, as described in the following picture.

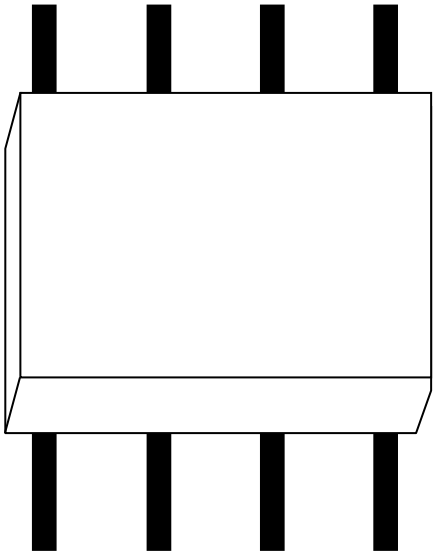


Write a program where the SENSE moves along the building and stops on the third black line.

The starting point of the SENSE should be on the other side of the building.

Challenge 3.12 – Taxi driver with passenger

Put black lines along the building, as described in the following picture.



Write a program where the SENSE moves along the building and stops on the third black line for 5 seconds.

After that, the SENSE continues to the other side and stops on the second black line.

Challenge 3.13 – Home vacuum cleaner robot

Build a model of a room as follows:

Create a program where the SENSE moves along the walls in different distances from the walls.

At the first round, the SENSE will move closer to the walls and at the second round, it will move 8 cm from the walls.