Université d'Ottawa
Faculté de Génie,
École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering
School of Electrical Engineering and
Computer Science

REAL TIME EMBEDDED SOFTWARE DESIGN

SEG 4145

LABORATORY EXPERIMENTS AND DOCUMENTATION

Winter 2014

Acknowledgements

We would like to thank, Alan Stewart for building the robots used for this course. The purchasing of Arduino boards were his selection.

Revision History

Version Number	Description	Date of Changes
1.0	The initial version of the lab manual was created by Dr. Dan Ionescu, in collaboration with Mohamad Forouzanfar and CEG students enrolled in CEG4912F of the academic year 2011-2012	Seprember 15, 2011
2.0	The second version of the lab manual was created by Raymond Peterkin	July 20, 2012
2.1	Corrections to the lab manual	January 6, 2013

Table of Contents

ACKNOWLEDGEMENTS ..... **I**

REVISION HISTORY ..... **II**

LABORATORY INSTRUCTIONS **1**

LAB GROUPS 1
PROGRAMMING LANGUAGE 1

DEMONSTRATION AND LAB REPORT **1**

DEMONSTRATION 1
LAB SUBMISSION 1
Laboratory Report *1*
Source Code *2*
MARKING SCHEME 3

DOCUMENTATION **3**

DEVELOPMENT TOOLS **4**

ROBOTIC PLATFORM 4
SOFTWARE 6

LAB PROCEDURES **7**

CHARGING THE ROBOT'S BATTERIES **7**

CREATING SOFTWARE PROGRAMS WITH THE ARDUINO IDE **8**

PROGRAM COMPILATION AND EXECUTION 8
MAIN STRUCTURE OF A PROGRAM 10
PROGRAMMING TASKS 11
Pin Manipulation *11*
Pulse Width Modulation *12*
LCD Screen Manipulation *12*
Sonar Sensor Manipulation *13*
Temperature Sensor Manipulation *14*
WI-FI MANIPULATION 14
EXAMPLE PROGRAMS 16
SERIAL MONITOR 17

LAB 1: INTRODUCTION TO THE ARDUINO IDE AND THE ROBOTIC PLATFORM **19**

LAB 2: PRECISE ROBOT MOVEMENT **21**

LAB 3: SENSOR DATA RETRIEVAL **23**

LAB 4: NETWORK COMMUNICATIONS **24**

LAB 5: GRAPHICAL USER INTERFACE DEVELOPMENT **26**

APPENDIX A: ROBOT TECHNICAL SPECIFICATIONS **27**

APPENDIX B: ROBOT MEASUREMENTS APPENDIX C: SOFTWARE COMMENTING SUGGESTIONS **29**

APPENDIX C: SOFTWARE COMMENTING SUGGESTIONS **30**

List of Figures

FIGURE 1 ROBOTIC PLATFORM FOR THE LABS ---------- 5
FIGURE 2 ARDUINO MEGA BOARD USED ON THE ROBOT ---------- 5

List of Tables

TABLE 1 LAB MARKING SCHEME 3

Laboratory Instructions

Lab groups

- Lab will be done in groups of two or three students depending on number of available robots.
- Students should remain in the same group and with the same TA during the entire semester.
- Lab report guidelines and the marking scheme will be explained in later sections of this manual.

Programming Language

- Labs will be developed using the arduino programming language.
- The development environment, recommended tools and instructions are described later in this manual.

Demonstration and Lab Report

Demonstration

- Each laboratory experiment will include one session. Ideally, you should perform a demonstration before the end of the session.
- The demonstration includes showing your source code and demonstrating a working system.
- If you cannot demonstrate your work before the end of the TA session, make an appointment to do so before the lab submission due date.

Lab submission

Every lab requires a report and all files to be submitted. The following subsections describe the details of these components.

Laboratory Report

- One lab report is required from each group.
- Reports are due two weeks from the date of performing the experiments.
- The report must be typed using 8 x 11.5" white paper and must be well fastened.
- Handwritten reports will NOT be accepted.
- Check for correct spelling and grammar
- Each report will contain a separate cover page, a list of all main and sub-objectives and a description of the experiment.

Cover Page
A cover page should include

- Title of the experiment.
- Course code and lab section
- Student names and numbers
- The date the experiment was performed

Content
A lab report should include the following.

- An explanation of your solution to the problem with flowcharts and diagrams wherever possible.
- A description of how separate modules (subroutines, interrupt service routines, etc.) link together.
- A section on any problems encountered and how they were solved.
- A discussion on whether your design worked or not and how it was tested to find out.
- A printout of all source code developed for the experiment. The source code should be included as an appendix.

While no specific format is required your report must include all of the elements that were just described. The following headings can be used as a guide for your report.

- Objectives
- High level design (list and describe modules and how they interconnect. Justify design decisions)
- Algorithms (descriptions of all algorithms using flowcharts)
- Discussion (did the design work, problems encountered, tests performed, etc.)
- Source code

Source Code

- All files must be properly commented.
    - The beginning of each file will contain a header with the following information:
        - Student names
        - Student numbers
        - Course code
        - Lab number
        - File name
        - General description of file.
    - Each function will contain a header with the following
        - Function name
        - Description of input variables if any (name, type, purpose)
        - Description of return variable if any (name, type, purpose)
        - Function description

- You may choose any format you wish so long as it conforms to the specifications listed above. Consult **Appendix C** for a suggested format to commenting files.

Marking Scheme

The following mark distribution will be used to access the labs.

Table 1 Lab marking scheme

Lab component	Overall percentage of lab mark
Demonstration	30%
Report	70%
Content	50%
Grammar	10% (2% deducted for each 'serious' error up to a maximum of 10%)
Source code (proper comments)	10%

Late submission of the lab report will result in a **10% deduction** per day up to a maximum of two days. **No reports are accepted more than two days late.**

Documentation

- Arduino Programming Language Reference
  - Official instruction guide
    - http://arduino.cc/it/Reference/HomePage

- C Programming References
  - Introduction to C Programming (a good introduction to C written by the University of Leicester)
    - http://www.le.ac.uk/cc/tutorials/c/
  - C Programming Tutorial (a fairly good, example based tutorial for beginners)
    - http://cplus.about.com/library/blctut.htm
  - Beej's Guide to C Programming (intended for UNIX users but very complete and well described)
    - http://beej.us/guide/c/src/bgc.html

Any reference that students wish to consult with C programming is fine. The references listed above are suggestions if students are very unfamiliar with C and need a place to start looking.

Development Tools

The sections below describe the tools that will be used in your lab experiments.

Robotic Platform

Laboratory experiments will be developed using robots like those shown in Figure 1. They were designed and built at the University of Ottawa to be used primarily by students in SEG 4145. Each robot has been built with the following components:

- One Arduino Mega 2560 board shown in Figure 2.
- Two batteries, located beneath the lower platform.
- Two motors located beneath the lower platform attached to wheels.
- One motor attached to a sensor array on top of the platform.
- One sonar sensor.
- One temperature array sensor.
- Two wheels, located in the bottom platform
- One LCD display, underneath the top platform.
- One LED light on the Arduino board.

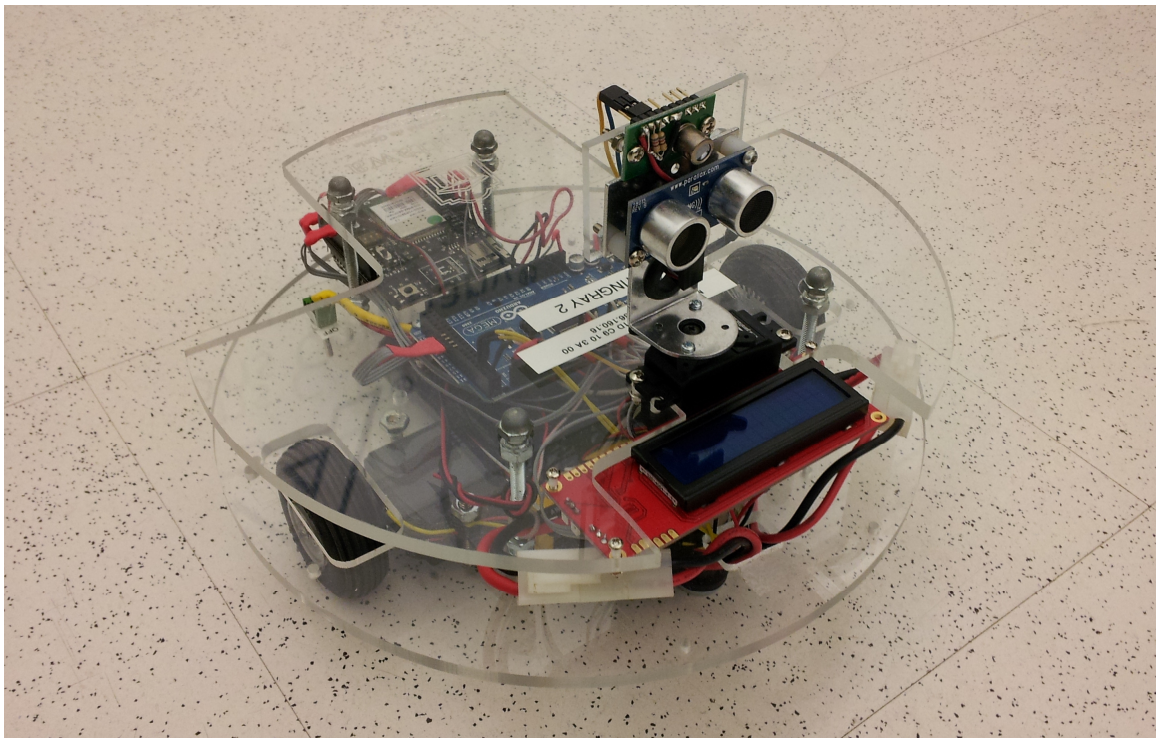


Figure 1 Robotic platform for the labs experiments/project

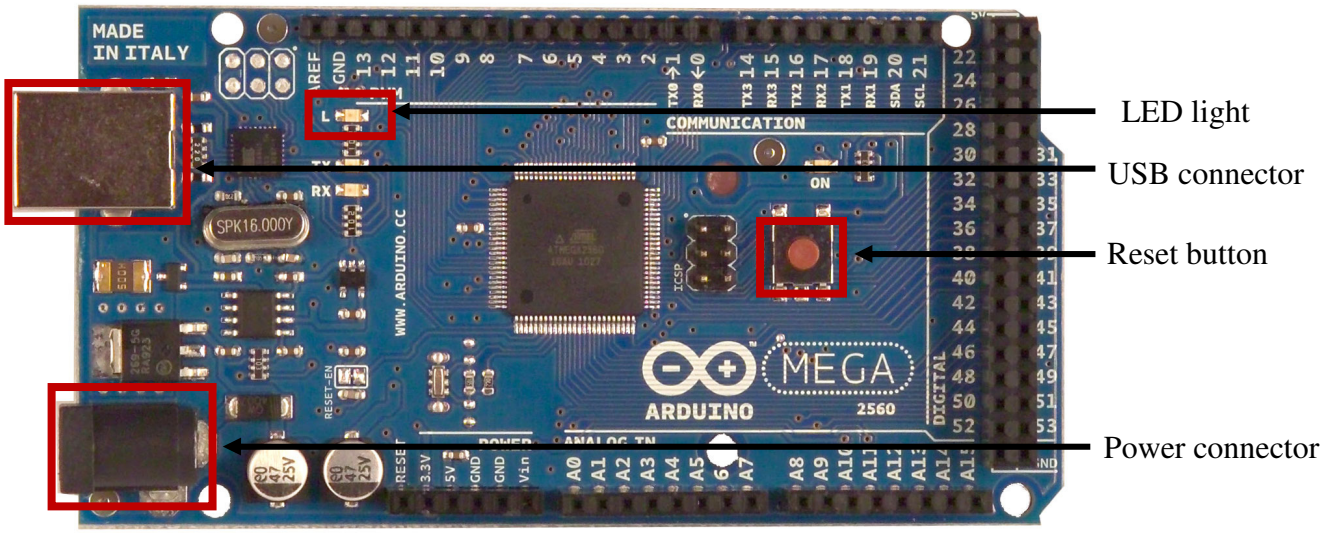


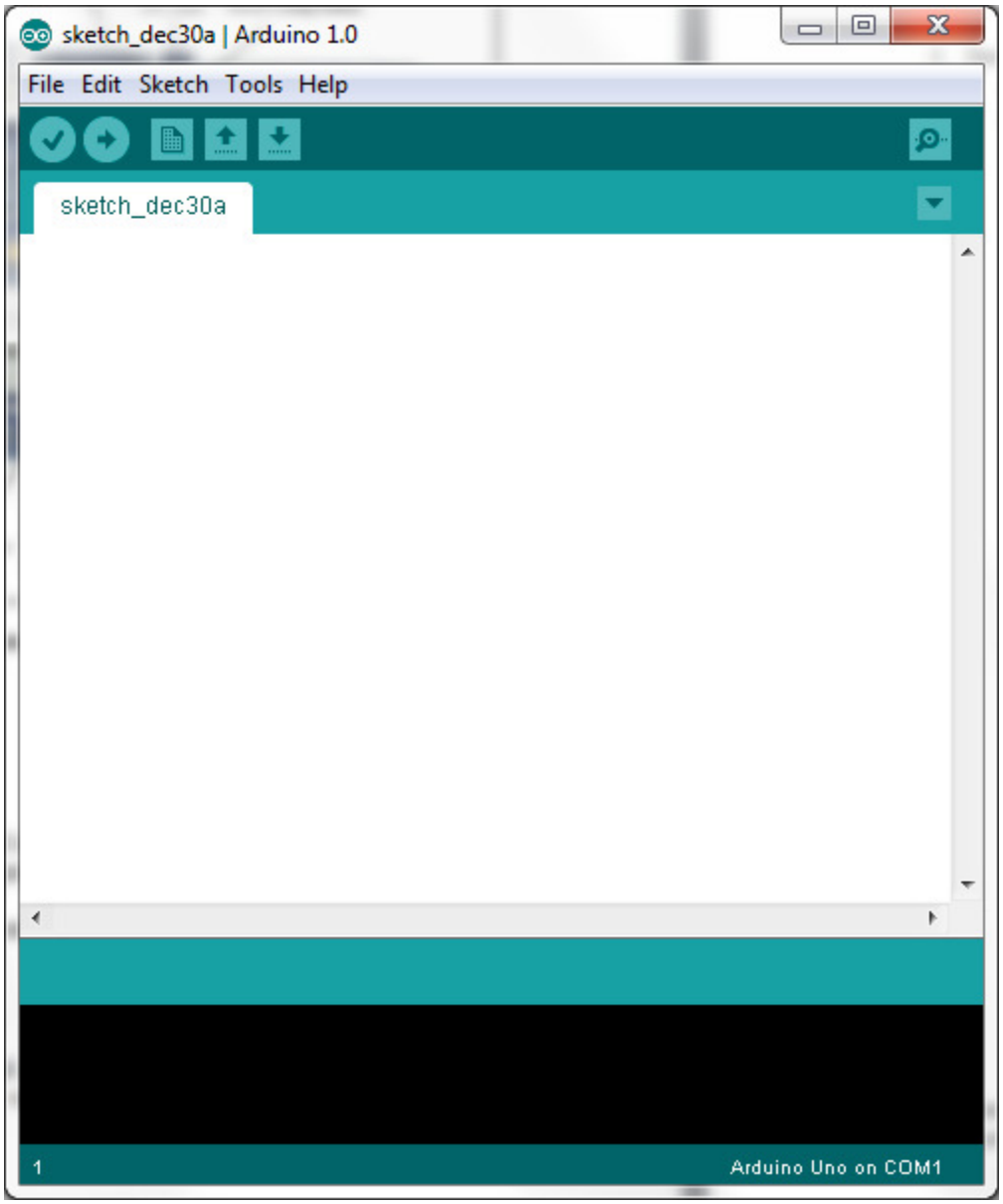
Figure 2 Arduino Mega board used for controlling the robot

Software

All software programs for the robot will be written, compiled and downloaded from a single program called the Arduino Integrated Development Environment (IDE).

The Arduino IDE can be started by selecting Start Menu→All Programs→EECS→Arduino→Arduino

A window similar to the following window should appear is the Arduino IDE has been started successfully:



Lab Procedures

When performing the laboratory experiments the following procedures should always be followed.

- Please make sure that the USB cable is connected **before** the robot is turned on. As a general rule all connections should be made before providing power to electronic devices.

- When you are finished working with the robot please turn it off **before** detaching the USB cable. For similar reasons to the previous point, connections should not be modified unless all power has stopped running through an electrical device.

- **Every program written should display the student numbers of both members** for at least a few seconds when it starts to ensure that it is not a program executed from flash memory belonging to another group.

Charging the Robot's Batteries

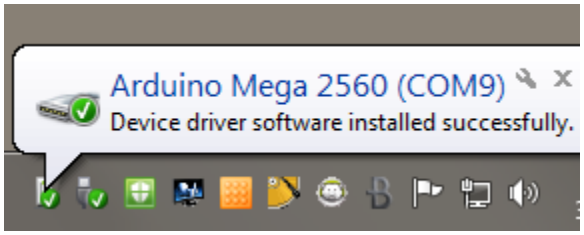
It is highly recommended that students use a power adaptor when testing their robots and use battery power for limited testing and demonstration purposes only to converse power. However, the batteries on the robot will eventually be depleted and need to be recharged. To charge the batteries, look for one of the two dual battery chargers in the LAB. The batteries should not be detached from the robot.

Creating Software Programs with the Arduino IDE

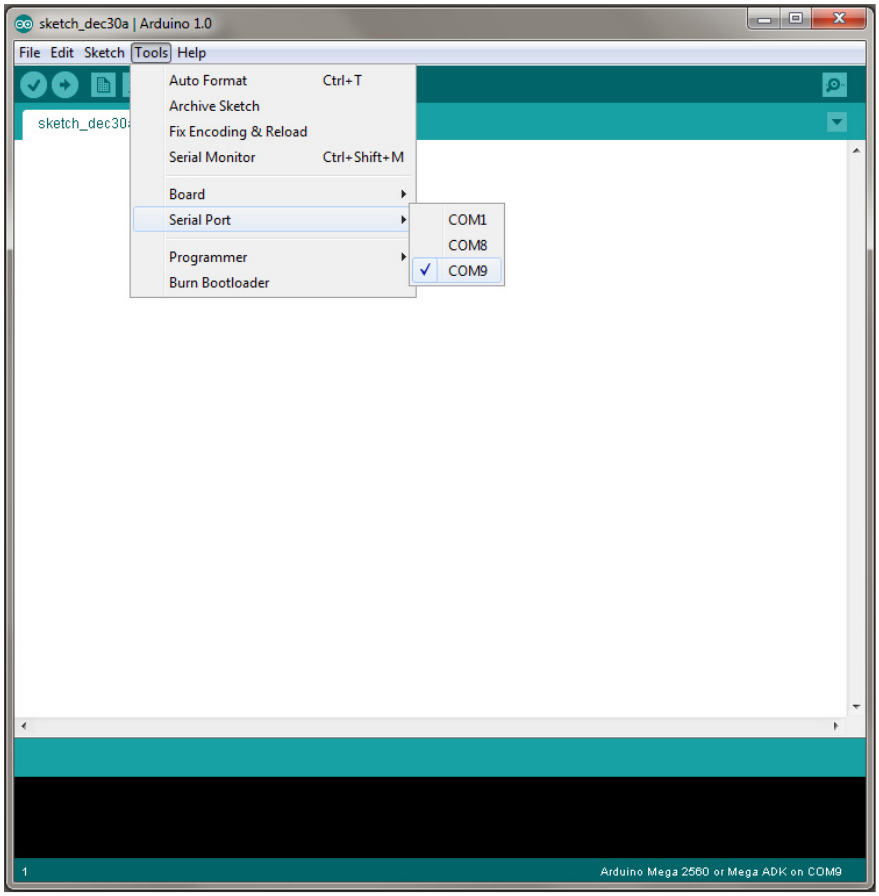
Program Compilation and Execution

When your program has been written it must be compiled and downloaded to the Arduino platform before execution can begin. The correct serial port and device must be configured in the Arduino IDE before a program can be compiled and downloaded. Perform the following steps for the robot to be recognized by the Arduino IDE:

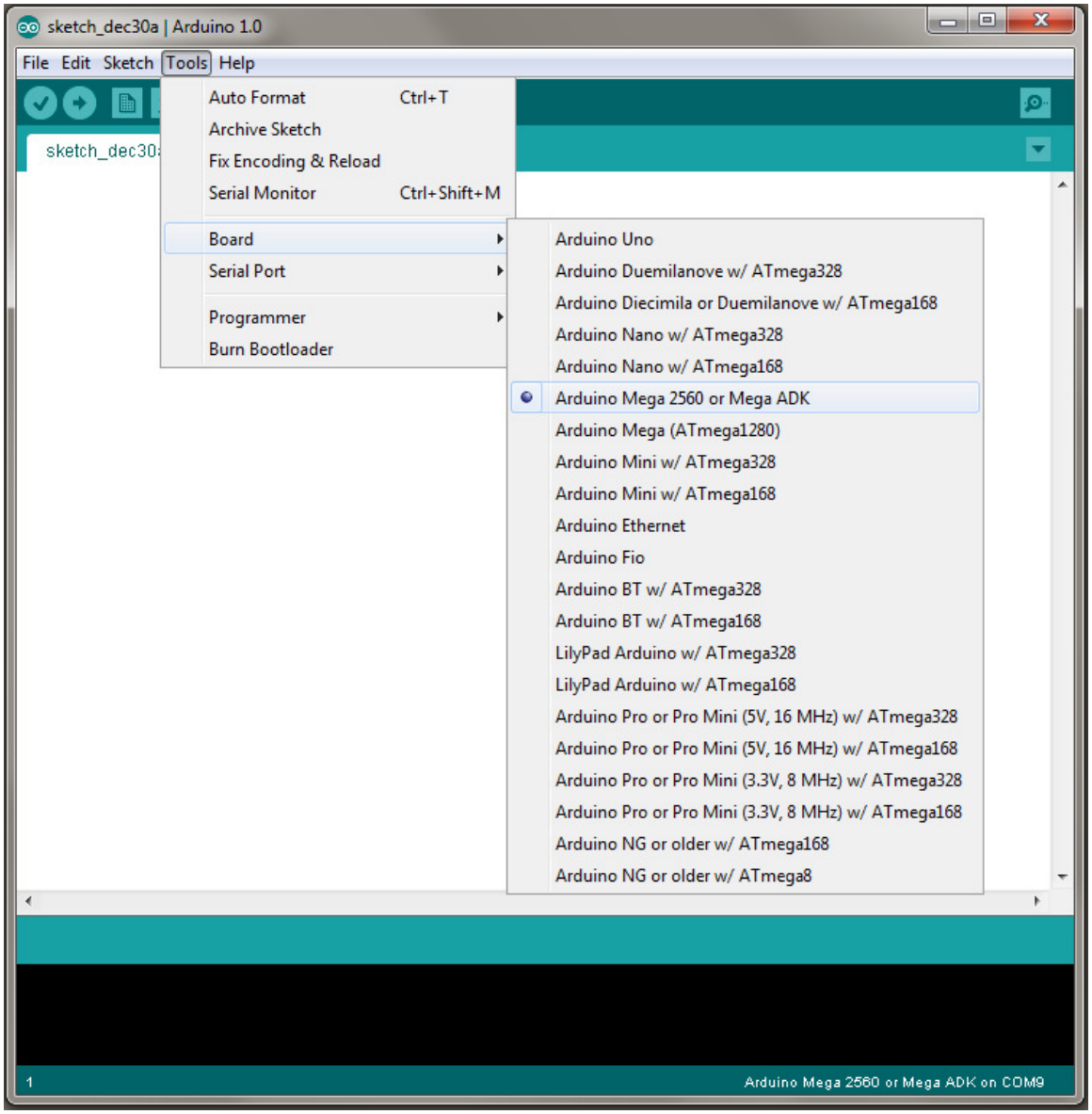
1. If you have not done so, connect the robot to the PC by connecting the USB cable attached to the PC to the USB connector of the robot. When the robot has been connected successfully a message indicating the COM port (i.e. COM9) that must be used with the robot. Make note of this COM port and proceed to the next step.



2. Open the Arduino IDE and select the COM port observed in step 1 by selecting **Tools→Serial Port** from the menu.



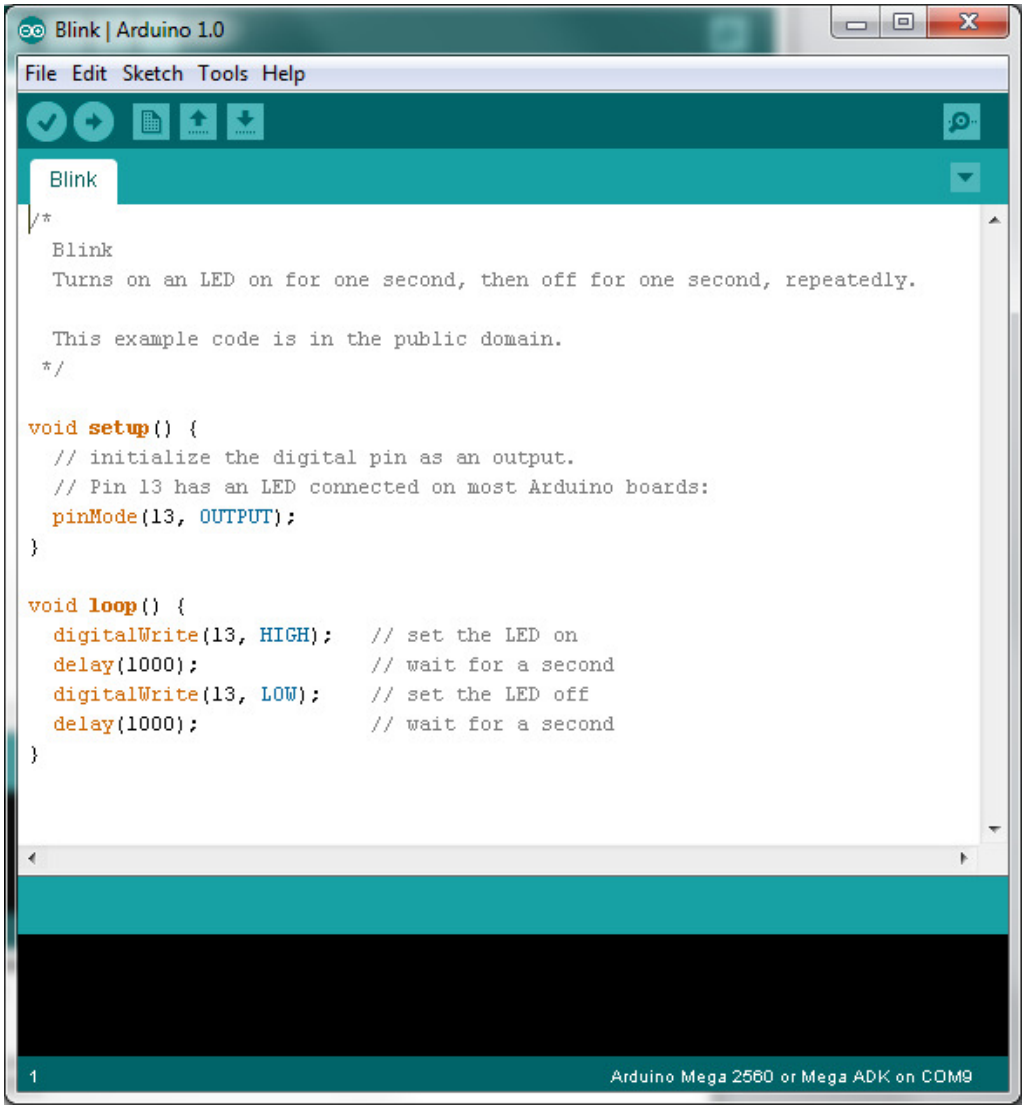
3. Select the ‘Arduino Mega 2560 or Mega ADK’ option from the **Tools→Board** menu.



The correct serial port and device should appear at the bottom of the Arduino IDE window.

To compile a program, select Sketch→Verify/Compile from the Arduino IDE window or click on the check mark (the left most icon in the menu) as shown in the window.

To upload a program to Arduino board, click on the arrow mark (the second on the left) as shown in the window.



Main Structure of a Program

A program written for the Arduino program takes the following general form:

```
// the setup routine runs once when you press reset:
// This function is called before the loop() function
void setup() {
   // Initialization instructions
}

// the loop routine runs over and over again forever:
void loop() {
```

```
  // The instructions executed to perform the main
  // functionality of the program
}
```

Each program must contain the following functions:
setup(): This function is used to initialize the Arduino program through a sequence of instructions that are executed one time.
loopO: This function is used to implement the main functionality of the program. This function is called endlessly to perform the execution of a program.

Other functions may be defined and called from the setup() and loop() functions as required.

Programming Tasks

Pin Manipulation

Connections are established with the Arduino platform through pins. Each pin is identified with a unique number and may be read for data retrieval or written for data transmission. Pins values may be HIGH or LOW. Pin values for all robot components can be found in Appendix A. The program demonstrated below illustrates the steps required for the initialization and use of pins to read data or write data:

```
// Global variables
int pin_value = 0;

void setup() {
  // Initialize pin 1 as an output pin.
  pinMode(1, OUTPUT);

  // Initialize pin 2 as an input pin
  pinMode(2, INPUT);
}

void loop() {
  // Write a LOW value to pin 1
  digitalWrite(1, LOW);

  // Write a HIGH value to pin 1
  digitalWrite(1, HIGH);

  // Read the value of pin 2
  pin_value = digitalRead(2);
  // pin_value can be compared to HIGH
```

```
  // or LOW to determine its value
}
```

The HIGH and LOW values for the wheel encoder indicate the angular rotation that has occurred with the respective wheel. Each transition of HIGH to LOW or LOW to HIGH indicates an angular rotation of 5.625 degrees. No difference between two consecutive values indicates that the wheel has not move.

It is important that there is as little delay as possible when reading successive values for the wheel encoders to ensure that the values remain consistent.

Pulse Width Modulation

Pulse width modulation is a technique used to control the power sent to devices like the motors on the robot. A voltage wave with specific characteristics must be transmitted to each motor in order for the motor to move in a specific direction. A wave is transmitted to a device using the following command:

```
int motor_pin_value;
int pulse_width;

analogWrite (motor_pin_value, pulse_width);
```

Consult Appendix A for motor pin values and the appropriate pulse width values.

LCD Screen Manipulation

A variable of the type 'SoftwareSerial' must be created to manipulate the LCD screen. The pin for the LCD screen must be specified with the initialization of this variable. The serial port must also be initialized in a similar manner to a serial port where a data transmission speed must be specified as its operation is started. An example of the variable creation and initialization is shown below:

```
#include <SoftwareSerial.h>

int lcd_pin_number = 18;
SoftwareSerial LCD = SoftwareSerial(0, lcd_pin_number);
LCD.begin(9600); // Open the serial port to
                 // write data at 9600 bps
```

Data is displayed on the LCD screen with the **print** function which is used in the following manner:

```
LCD.print("Hello world!");
```

Commands are single integer values written with the **write** function. The LCD screen must be put into a command execution mode before the desired instruction can be executed. It is also highly recommended that a small delay is executed after in case numerous instructions are executed to limit the amount of data transmitted to the LCD screen at one time. An example of the instructions required to clear the LCD screen are demonstrated below:

```
LCD.write(0x7C);  // Put the LCD screen in command mode.
LCD.write(0x01);  // Clear the LCD screen.
delay(10);        // Delay operation for 10 ms
```

The following commands must be used to position the cursor at a specific location on the LCD screen:

```
LCD.write(0x7C);  // Put the LCD screen in command mode.
LCD.write(col + row*64 + 128);  // Place the cursor in the
position (row,col).
delay(10);        // Delay operation for 10 ms
```

The row and column of the cursor are denoted by the variables row and col respectively. The top and bottom rows of the LCD screen are described with the numbers 0 and 1 respectively. Each row of the LCD screen is 16 characters wide with each position denoted with the numbers 0 through 15 inclusive.

A full list of commands for the LCD screen can be found in Appendix A.

Sonar Sensor Manipulation

The sonar sensor may be used to determine the distance between the robot and the nearest object in the direction of the sensor's orientation. The following procedure must be followed in order to determine the distance between the sonar sensor and the nearest object:

1. Set up the sonar sensor pin as an output pin.
2. Write a low value to the pin for two microseconds.
3. Write a high value to the pin for five microseconds.
4. Write a low value to the pin.
5. Set up the sonar sensor pin as an input pin.
6. Read the duration of the pulse from the sonar sensor pin (in microseconds) using command pulseIn(pin, HIGH).
7. Convert the pulse duration time to distance using the following equation:
    a. Distance = time / (29 * 2)
    where time is in microseconds and distance in centimeters. This equation is based on the speed of sound which is 340 m/s. The pulse must travel to the object and back so half the distance value is considered.

Temperature Sensor Manipulation

The temperature sensor of the robot generates the following values:

- Ambient (room) temperature: The temperature of the immediate surroundings
- Eight adjacent temperature values directly in front of the sensor.

All temperature values are valid within approximately two metres.

The temperature sensor is connected to the I2C bus. The following code demonstrates how data can be read from the temperature sensor using the I2C bus.

```
#define TEMPSENSOR 0x68
#include <Wire.h>

int reg = 0x01; // This register value determines if the
                // ambient temperature will be read or one
                // of the other eight temperature values
                // Ambient temperature = 0x01
                // 8 arrays values = 0x01 to 0x09


void setup() {
    Wire.begin();  // Join the I2C bus
}

void loop() {
    Wire.beginTransmission(TEMPSENSOR);
    Wire.write(reg); // Indicate temperature value to read
    Wire.endTransmission();
    Wire.requestFrom(TEMPSENSOR, 1);  // Request data
    while(Wire.available() < 1);  // Wait for data
    byte temperatureData = Wire.read(); // Temp. value
    delay(50); // Delay 50 ms if values are read in a loop
}
```

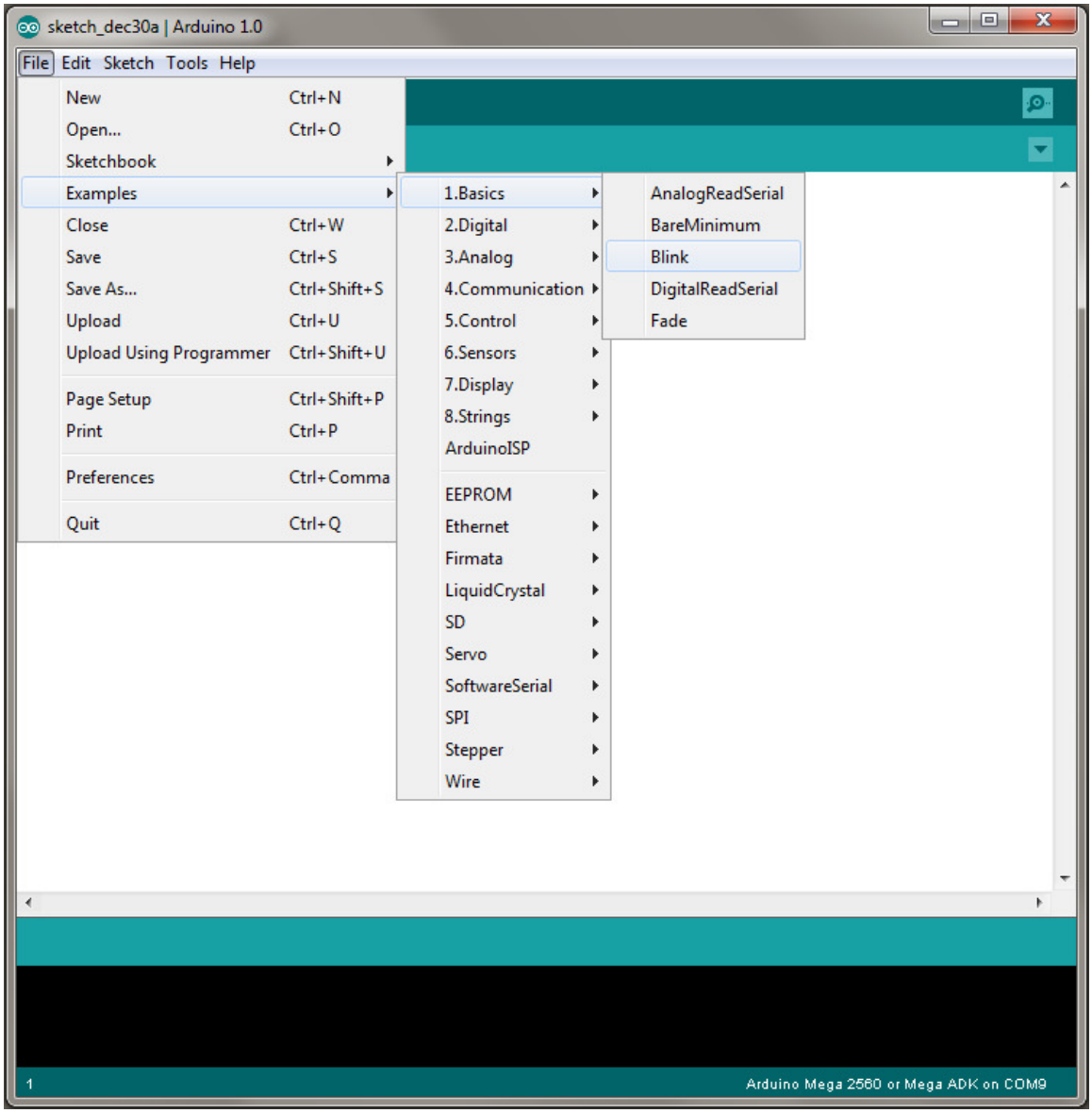
Wi-Fi Manipulation

A Wi-Fi module has been installed on the robot to permit wireless connections to a PC or other robots for network based communication. Each robot must establish a connection to an existing Wi-Fi network before data can be read and processed. The code below illustrates an example of how a connection can be established with a Wi-Fi network using, Dynamic Host Configuration Protocol (DHCP), meaning that an IP address is automatically assigned to the robot for communication with other devices:

```
#include <Wirefree.h>
#include <WifiServer.h>

WIFI_PROFILE w_prof = { "networkname",    /* SSID */
                        "wifi_password" , /* passphrase */
                        NULL,             /* DHCP */
                        "255.255.255.0" , /* subnet mask */
                        "192.168.0.1"   , /* Gateway IP */
                      };

// port 80 is default for HTTP
WifiServer server(80);

void parseRxData(String data)
{
}

void setup() {
     // connect to AP and start server
    Wireless.begin(&w_prof, &parseRxData);
    server.begin();

    delay(1000);
}

void loop()
{
  // Listen for incoming clients
  WifiClient client = server.available();
  if (client) {
    // an HTTP request ends with a blank line
    while (client.connected()) {
      if (client.available()) {
        int  b;

        while((b = client.read()) == -1);
        // Process data

      }
    }
    // close the connection
    client.stop();
  }
}
```

Example Programs

Numerous examples are available to assist students in developing various aspects of Arduino programs. These programs can be accessed from the Arduino IDE by selecting File→Examples and selecting the correct sub menu item as illustrated below:

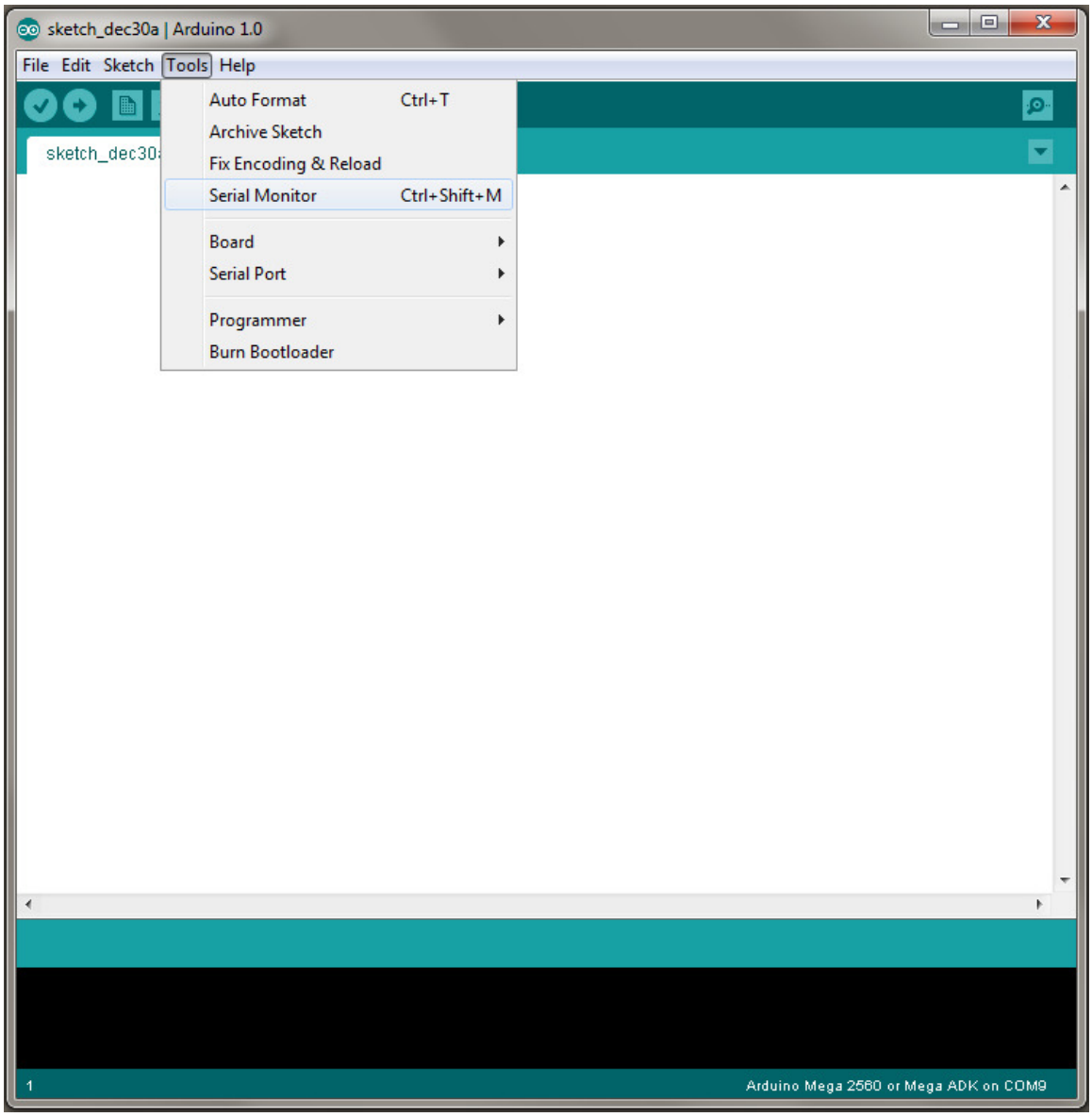


Serial Monitor

The serial monitor permits Arduino program developers to print statements to a console as their program is executing. This feature provides a real-time debugging allowing for statements to be displayed instructions are executed. Statement must be written to the serial port for them to be displayed properly through the serial monitor. The following program will print one statement to the serial monitor every second during the execution of the program:

```
// Global variables
int i = 1;

void setup() {
  // Open the serial port at 9600 bps
  Serial.begin(9600);
}

void loop() {
   Serial.print("Line number: ");
   Serial.print(i);
   Serial.println();
   delay(1000);
}
```

The serial monitor must be executed after the program has been verified and transferred to the Arduino platform for successful execution. Select Tools→Serial Monitor from the Arduino IDE as shown in the figure below:


sketch_dec30a | Arduino 1.0
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
1
Arduino Mega 2560 or Mega ADK on COM9

Lab 1: Introduction to the Arduino IDE and the Robotic Platform

Objectives:

On completion of this lab, students will be able to perform the following tasks:

- Implement and download a software program for the robot using the Arduino IDE.
- Manipulate the LCD display, LED light and wheels of the robot.
- Manipulate the LCD display of the robot.
- Manipulate the LED light of the robot.
- Manipulate the wheels of the robot to move the robot forwards, backwards and rotate the robot clockwise and counterclockwise.

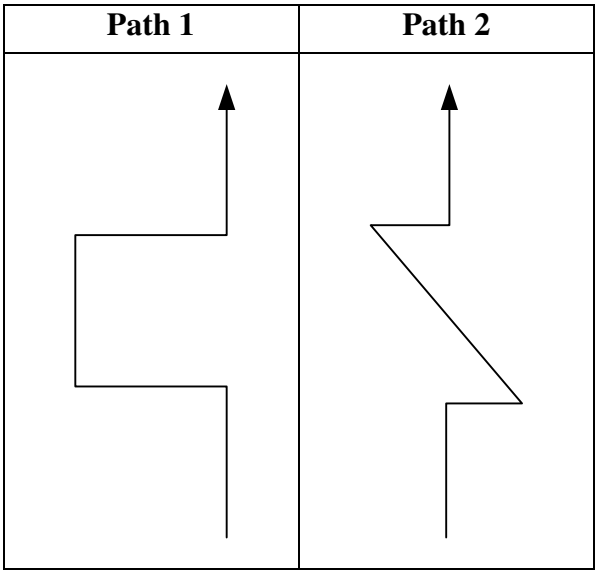
Tasks:

First write a program that continuously makes the robot rotate in one direction (either clockwise or counter clockwise). Time how long it takes for the robot to perform a full rotation and use that value to approximate angular rotations. For example, if it takes 10s for a rotation, then a 360° takes 10s, 180° takes 5s, 90° takes 2.5s, etc. This step is being performed since all motors are different so the rotation times for each robot might be slightly different.

HINT: Your robot can be made to 'rotate' by moving the wheels in opposite directions at the same time.

Write a program for the robot that performs the following tasks:

- Display the two student numbers of the students for 5 seconds and have the robot traverse the following paths described below:



Note: Each line segment should take 10 seconds to travel. The rotations in paths 1 are 90°. All rotations in paths 2, are 45° (or the appropriate multiple of 45°) from the horizontal.

The following characteristics must be exhibited on the robot as the program is operating:

- Display one student number on each line of the LCD display at the beginning of the program for 5 seconds.
- As the student numbers are being displayed, the LED light must flash/flicker after each second to indicate the length of time that has passed
- The message "Path 1" and "Path 2" must be displayed on the top line of the LCD screen for 3 seconds before the respective path is traversed. The bottom line of the LCD screen must be blank.
- All text displayed on the LCD display must be centered on the top and bottom lines.
- The following messages must be displayed on the LCD display as the robot makes its various movements:
    - moving forward
    - moving backward
    - rotating left
    - rotating right
    - stopped

Each message with two words must display the first word on the stop line and the second word on the bottom line. A message with one word must be displayed entirely on the stop line with a blank bottom line.

Your program should be as modular as possible. The following steps are **highly** recommended in the development of this lab:

- Your program should be as modular as possible. The following steps are **highly** recommended in the development of this lab:
    - Defining macros in the header file for both motors and all possible directions (i.e. one macro for when the left motor is stopped, one macro for when the right motor rotates backwards, etc.)
    - Defining functions to stop the robot, rotate it clockwise or counterclockwise, and moving it forward and backwards.

- **BONUS:** After traversing a path, the robot must pause for approximately 5 seconds, and traverse the same path **backwards**, ending up in approximately the same position it started. 2.5% will be added for every path that can be correct traversed backwards for a maximum bonus of 5% overall.

Lab 2: Controlled Robot Movement

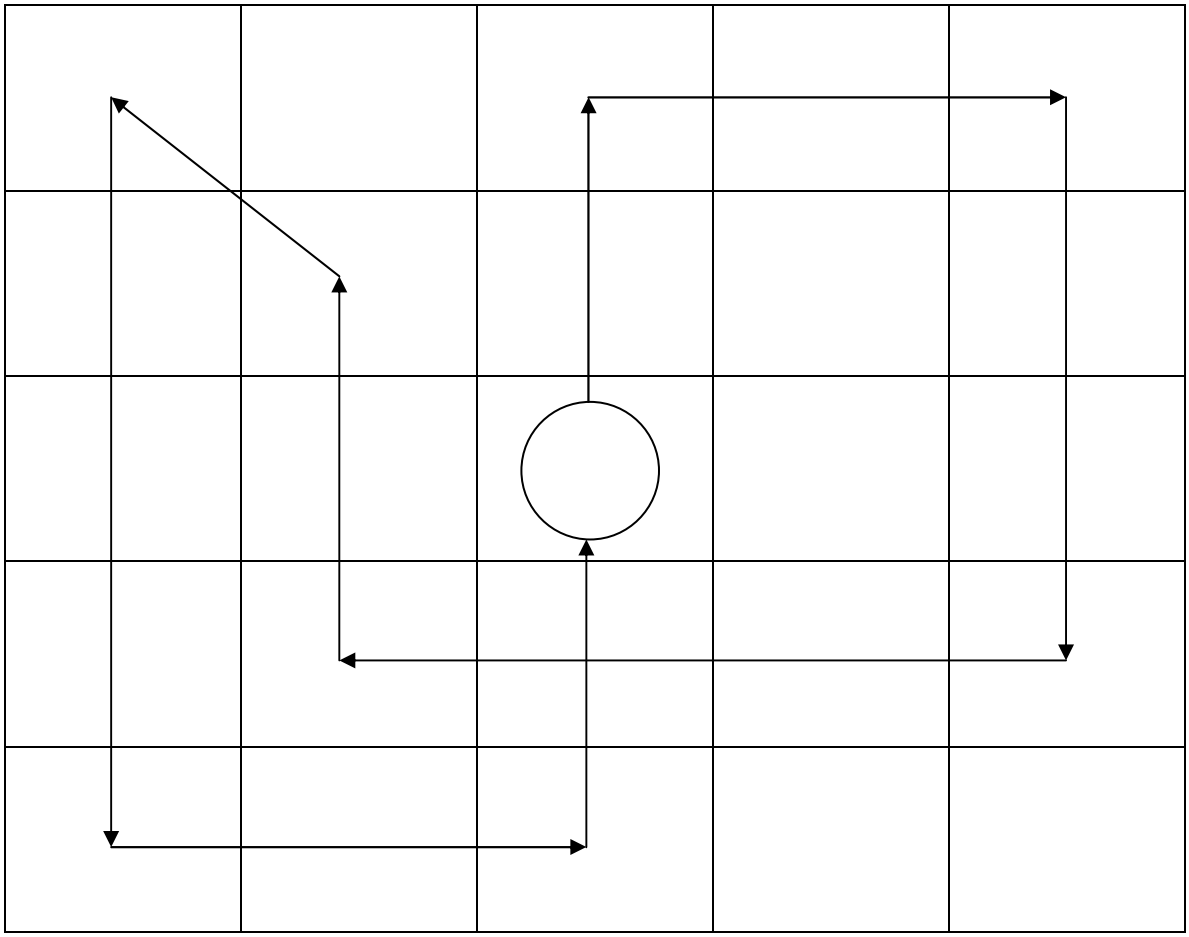
Objectives:

On completion of this lab, the student will be able to perform the following tasks:

- Process sensor data by reading and processing the wheel encoder data in real-time.
- Control the movements of the robot in a precise manner.

Lab contents:

Using the information in earlier sections of this lab and from Appendix A write a program for the robot to traverse the following path:



Each square in the grid represents one tile on the ground that must be traversed by the robot. Consult Appendix B for the measurements required to traverse the path with

precision. The following characteristics must be exhibited on the robot as the program is operating:

- Display one student number on each line of the LCD display at the beginning of the program for 5 seconds.
- As the student numbers are being displayed, the LED light must flash/flicker after each second to indicate the length of time that has passed.
- All text displayed on the LCD display must be centered on the top and bottom lines.
- The following messages must be displayed on the LCD display as the robot makes its various movements:
    - moving forward
    - moving backward
    - rotating left
    - rotating right
    - stopped

Each message with two words must display the first word on the stop line and the second word on the bottom line. A message with one word must be displayed entirely on the stop line with a blank bottom line.

- Your program should be as modular as possible. The following steps are **highly** recommended in the development of this lab:
    - Defining macros in the header file for both motors and all possible directions (i.e. one macro for when the left motor is stopped, one macro for when the right motor rotates backwards, etc.)
    - Defining functions to stop the robot, rotate it clockwise or counterclockwise, and moving it forward and backwards.

BONUS: After traversing a path, the robot must pause for approximately 5 seconds, and traverse the same path **backwards**, ending up in approximately the same position it started. A maximum bonus of 5% may be added at the discretion of the TA evaluating lab demonstration.

Lab 3: Sensor Data Retrieval

Objectives:

On completion of this lab, students will be able to perform the following tasks:

- Process distance and temperature data on the robot in real-time.
- Implement collision avoidance algorithms based on real-time data.

Lab contents:

Write a program for the robot to perform the following tasks:

- The robot travels forward in a straight line one tile at a time stopping for one second after one tile has been traversed. If an object has been detected approximately 10 cm away or closer the following collision avoidance algorithm must be performed:
    - The robot must come to an immediate stop
    - The ambient temperature must be measured by the robot and displayed on the LCD screen for 5 seconds with the following message:
        - Temperature (top line)
        - x degrees (bottom line)
    - The robot must move backwards one tile.
    - The robot must rotate clockwise approximately 90 degrees
    - The robot must move forward two tiles
    - the robot must rotate counter clockwise approximately 90 degrees
    - The robot continues moving forward until another object is encountered
- It can be assumed that no other object will be countered as the robot is performing the collision avoidance algorithm.

The following characteristics must be exhibited on the robot as the program is operating:

- Display one student number on each line of the LCD display at the beginning of the program for 5 seconds.
- As the student numbers and temperature data are being displayed, the LED light must flash/flicker after each second to indicate the length of time that has passed.
- All text displayed on the LCD display must be centered on the top and bottom lines.
- The following messages must be displayed on the LCD display as the robot makes its various movements:
    - moving forward
    - moving backward
    - rotating left
    - rotating right
    - stopped

Lab 4: Network Communications

Objectives:

On completion of this lab, the student will be able to

- Establish a connection to a Wi-Fi network using the robot.
- Use a PC to exchange data with the robot and execute various commands in real-time.

Lab contents:

First write a program for the robot to establish a Wi-Fi connection with the network in the lab. Use the information described earlier in this manual to establish a DHCP connection.

Write a console program using the Java programming language that automatically establishes a connection with the robot and performs the operations described in the following menu options:

```
Enter the correct number to select an operation:
1 – Move the robot forward.
2 – Move the robot backward.
3 – Rotate the robot clockwise.
4 – Rotate the robot counter clockwise.
5 – Read the distance to the nearest object.
6 – Read temperature values.
7 – Quit.
```

An error message should be displayed if a connection cannot be established with the robot and the program should immediately terminate. Selecting option 1 or 2 must prompt the user to enter the distance for traversal in centimeters from 0 and 20 centimeters inclusive. Selecting options 3 or 4 must prompt the user to enter the rotation value in degrees from 0 to 359 inclusive. Selecting option 6 must prompt the user to return all temperature values that can be generated from the robot.

Write a program for the robot to perform the tasks described for the Java program with the following characteristics:

- Display one student number on each line of the LCD display at the beginning of the program for 5 seconds.
- As the student numbers are being displayed, the LED light must flash/flicker after each second to indicate the length of time that has passed.
- All text displayed on the LCD display must be centered on the top and bottom lines.

- The following messages must be displayed on the LCD display as the robot makes its various movements:
    - moving forward
    - moving backward
    - rotating left
    - rotating right
    - stopped

The robot must stop automatically after performing any movement or rotation.

A communications protocol must be designed and implemented to transmit data successfully between the robot and the Java program. The design and implementation of this protocol is left to the discretion of the students. This protocol may be implemented as simply as possible with no need for error checking or data verification mechanisms.

The communications protocol must be fully documented in your lab reports.

Lab 5: Graphical User Interface Development

Objectives:

On completion of this lab, the student will be able to perform the following tasks:

- Control the operations of an autonomous robot in real-time using a graphical user interface.

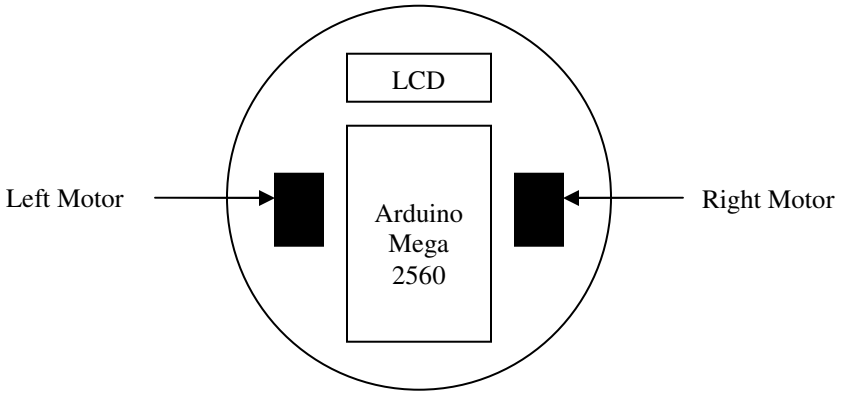
Lab contents:

- Replace the console application developed in lab 4 with a graphical user interface for manipulating the movements of the robot. The graphical user interface must be implemented with the following requirements:
    - Every operation performed in the console application must be performed through the graphical user interface.
    - Every movement and rotation of the robot must be documented with a timestamp (date and time) generated by the graphical user interface.
    - Distance and temperature values must be documented with a timestamp (date and time) generated by the graphical user interface.
- The program written for the robot in lab 4 must be used for lab 5.

BONUS: Additional marks will be awarded for graphical user interfaces employing a graphical method (i.e. a grid instead of a table) to describe the movements and rotations of the robot. A maximum bonus of 5% may be added at the discretion of the TA evaluating lab demonstration.

Appendix A: Robot Technical Specifications

The diagram below provides a high level illustration of the robot for the purposes of describing its specifications:



The table below lists the pin numbers for all the devices connected to the Arduino platform and the possible operations for each pin.

Device	Pin Number	Operation
Left motor	2	Write
Right motor	4	Write
Centre (sensor) motor	7	Write
Arduino Board LED	13	Write
LCD display	18	Write
Sonar sensor	22	Read
Left wheel sensor	26	Read
Right wheel sensor	28	Read

The table below illustrates common commands that can be used to manipulate the LCD screen:

Command	Value (Hex)
Clear Screen	0x01
Turn off the display	0x08
Turn on the display	0x0C
Scroll text to the left	0x18
Scroll text to the right	0x1C
Turn off back light	0x80
Turn on back light	0x9D
Command flag	0xFE

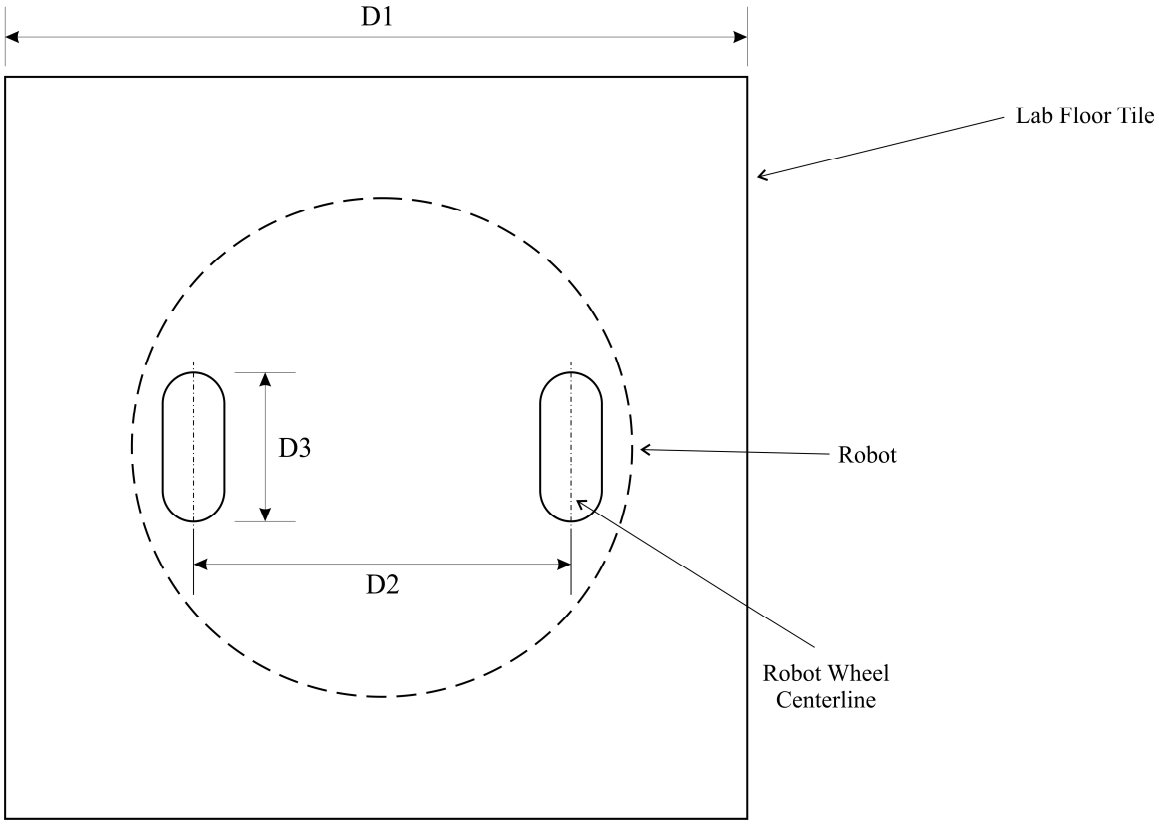
The table below illustrates the pulse width values required to manipulate the motors:

Motor	Direction	Pulse Width Value
All Motors	Stop	0
Centre Motor	Left	191.5
Centre Motor	Right	10
Left Motor	Backward	10
Left Motor	Forward	191.5
Right Motor	Backward	191.5
Right Motor	Forward	10

Appendix B: Robot Measurements

Some of the experiments will consist of having the robot moving in a precise manner. It might be useful to know some distances in order to precisely calculate paths. Using the figure below as reference, the following distances are given:

- D1 (Lab Floor Tile Width/Height) = **30.5 cm**
- D2 (Distance between Wheels) = **20 cm**
- D3 (Wheel Diameter) = **5.5 cm**



Appendix C: Software Commenting Suggestions

The following templates are to be included in source code:

- File template block: Placed at the beginning of every file to indicate its purpose.
- Function template block: Placed above every function to describe its inputs, outputs and purpose.

File template block:

```
/*****************************************************************************
 *
 * Names:          John Smith  1234567
 *                 Jane Doe    4567890
 * Course Code:    SEG 4145
 * Lab Number:     1
 * File name:      filename.
 * Date:           January 1, 2013
 *
 *
 * Description
 * ************
 * A simple description of the file goes here
 *
 *
 *****************************************************************************
 */
```

Function template block:

```
/*****************************************************************************
 *
 * Name
 **************
 * functionName
 *
 *
 * Description
 * **************
 * The description goes here
 *
 *
 * Parameters
 * **************
 * Name          Type          In/Out          Description
 *-----------    ----------    ---------------  ---------------
 *x             int           In              Description of x variable
 *y             char          In              Description of y variable
 *z             double        In              Description of z variable
 *
 *
 * Returns
 * **************
 * Name          Type          Description
 *-----------    ----------    ---------------
 *a             int           Description of a
 *
 *
 *****************************************************************************
 */
```