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 AND EMBEDDED SOFTWARE DESIGN

SEG 4145

LABORATORY 4
Network Communications

Winter 2019

Lab 4: Network Communications

Objectives:

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

- Establish a connection to a WiFi 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 WiFi connection with the network in the lab. Use the information provided at the end of this manual to establish a 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.

Installing the Wirefree Library

The Hydrogen WiFi Shield from DIY Sandbox has been installed on the robot to permit wireless connections to a PC or other robots for network based communication.
The Wirefree library should be installed in order to fully control the Hydrogen WiFi shield:

1. Download and install the Wirefree library. You can obtain it from the Course Portal. It is in a zip file under the SEG4145_Lab_Manual_04_Lab4.pdf file.
   You can also get it from github. For this purpose, complete the following steps:
   a. Go to the site https://github.com/diysandbox/Wirefree/
   b. Download the zip file (Wirefree-master.zip)
   c. Extract the zip file to the /Libraries directory of the arduino installation and rename it to ensure there are only alphanumeric characters:
      i. Go to H:\Arduino\libraries
      ii. Extract the zip file into the 'libraries' folder
      iii. Rename the folder so it only contains alpha numeric characters (i.e. WirefreeMaster)
2. Start arduino and verify that a folder with the library name exists under the examples option in the File menu (File-->Examples-->WirelessMaster)

WiFi Manipulation

Each robot must establish a connection to an existing WiFi network before data can be read and processed. The code below illustrates an **example** of how a connection can be established with a WiFi network based on the User Datagram Protocol (UDP):

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

WIFI_PROFILE wireless_prof = {
                        /* SSID */ "Robolab",
          /* WPA/WPA2 passphrase */ "w1r3l3ss!",
             /* Robot IP address */ "XX.XXX.XXX.XX",
                  /* subnet mask */ "255.255.255.0",
                   /* Gateway IP */ "10.136.160.1", };

String remote_server = "YYY.YYY.YY.YYY"; // peer device IP
address.
```

```
String remote_port = "9876"; // arbitrary
//Creates a client that can connect to a specified internet
IP address and port number
WifiClient client(remote_server, remote_port, PROTO_UDP);

void setup()
{
  // connect to AP
  Wireless.begin(&wireless_prof);

 // if you get a connection, report back via serial.
client.connect() connect to the IP address and port
specified earlier. It returns true if the connection
succeeds, false if not.
  if (client.connect()) {
    Serial.println("connected");  // prints to serial
monitor. Check the Serial Monitor Section at the end of
this manual.

    // Send message over UDP socket to peer device
    client.println("aBcDe"); //Your own message
  }
  else {
    // if connection setup failed:
    Serial.println("failed");
  }
}

void loop()
{
  // if there are incoming bytes available from the peer
device, read them and print them:
  while (client.available()) {
    int in;

    while ((in = client.read()) == -1);

    Serial.print((char)in);
  }

  delay(1);
}
```

Note: Choice of the communication protocol is up to the students.
Note: More examples can be found in File-->Examples-->WirelessMaster. You need to modify the default information under the "**WIFI_PROFILE w_prof ={ } ;**" section of

the sketch based on the network information given at the end of this manual and the IP address of the robot written on it.
Now, complete the following steps:

1) Connect the Arduino with attached Hydrogen Shield to your computer via USB cable.

2) On the Hydrogen, move the **WIFI UART: SW/HW** switch to **SW** by pressing it to the **left**. This switch is **S3** on the Hydrogen; it is located on the top right corner of the PCB, between the headers and the WiFi module, nearest the LEDs.)

   Note: The Hydrogen Shield communicates with the Arduino through UART lines. Unfortunately, when you program your Arduino, it also communicates through the UART lines; so switching the Hydrogen Board into Software UART will temporarily disable the WiFi radio while the unit is being programmed.

3) Once the upload process has completed successfully, move the Hydrogen **WIFI UART: SW/HW switch (S3)** back to **HW** by pushing the switch to the **right**.

4) Restart the Hydrogen by pushing the **Restart Button (S2)** one time.

5) The LED will light up solid green once the network has been created. Your Stingray Robot is now WiFi Capable.

   Note: This may take several minutes when connecting for the first time.

Robot Network Information

Use the following information to setup any communications:

SSID: Stingray
Password: w1r3l3ss!
Subnet Mask: 255.255.255.0
Gateway: 10.136.160.1
Security type: WPA-Personal
Encryption type: TKIP

You can find the robot IP and MAC address on your own robot.

If the password does not work, please, go and ask the Lab Supervisor Mr. Alan Stewart if there are some modifications to the above, as the WiFi router is in his room and under his control

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, or the serial monitor icon on the right side of the Arduino IDE, as shown in the figure below: