Stensat Group LLC, Copyright 2018

Legal Stuff

- Stensat Group LLC assumes no responsibility and/or liability for the use of the kit and documentation.
- There is a 90 day warranty for the Quad-Bot kit against component defects. Damage caused by the user or owner is not covered.
    - Warranty does not cover such things as over tightening nuts on standoffs to the point of breaking off the standoff threads, breaking wires off the motors, causing shorts to damage components, powering the motor driver backwards, plugging the power input into an AC outlet, applying more than 9 volts to the power input, dropping the kit, kicking the kit, throwing the kit in fits of rage, unforeseen damage caused by the user/owner or any other method of destruction.
- If you do cause damage, we can sell you replacement parts or you can get most replacement parts from online hardware distributors.
- This document can be copied and printed and used by individuals who bought the kit, classroom use, summer camp use, and anywhere the kit is used. Stealing and using this document for profit is not allowed.
- If you need to contact us, go to www.stensat.org and click on contact us.

References

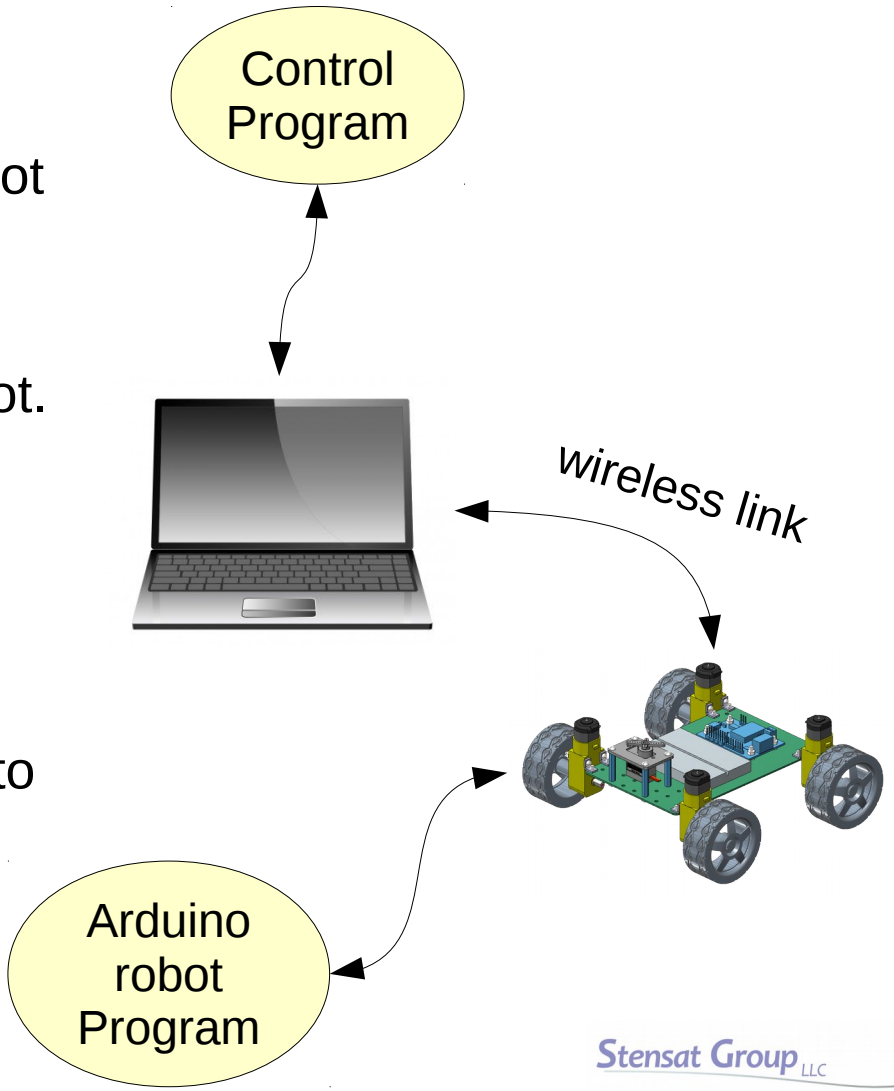
- www.arduino.cc
- http://esp8266.github.io/Arduino/versions/2.1.0/doc/reference.html

Introduction

- In this lesson, you will learn how to control the robot motion over WiFi.
- This will require writing an arduino program and a processing program.
- Review the WiFi lesson from the SLATE.

System Architecture

- This drawing shows how everything is interconnected.
- The control program runs on the laptop.
- The laptop wifi connects to the robot wifi.
- The control program sends commands over the wifi to the robot.
- The arduino program on the robot interprets the commands and executes them.
- The arduino program on the robot also sends telemetry over the wifi to the control program on the laptop.



What is the Plan

- In this lesson, you will learn how to program the robot to set up the WiFi as an access point and receive commands to move from your computer.

Wifi Configuration

- First thing to do is include the **ESP8266WiFi** library by adding the include statement to the top of the program.

```
#include <ESP8266WiFi.h>
```


Stensat Group LLC

Wifi Configuration

- Some items need to be declared.
- A **WiFiClient** object needs to be created. This allows the code to get commands from the laptop and send telemetry.
- **WiFiServer** object needs to be created so the laptop can connect and and send data to the robot. This allows the robot to receive connections.
- When creating the **WiFiServer** object, the network port is selected.

```
#include <ESP8266WiFi.h>

WiFiClient client;
WiFiServer server(80);
```

Wifi Configuration

- A character array is created for holding the commands sent by the laptop.
- For now, the first character in the array will be the command.

```
#include <ESP8266WiFi.h>

WiFiClient client;
WiFiServer server(80);

unsigned char cmd[6];
```


Stensat Group LLC

Wifi Configuration

- In the **setup()** function, the digital pins controlling the motors is initialized to be all outputs.
- WiFi.mode(WIFI_AP) configures the WiFi to operate as an access point allowing laptops and other clients to connect to it. When in this mode, any client that connects will be given an IP address. Up to 4 clients can connect.
- The WiFi is set up as an access point. **WiFi.softAP()** will set up the robot as an access point with the SSID specified. If a password is desired then the format is:
  - **`WiFi.softAP("ssid","password");`**
- After the access point is configured, the server is started. This implements the ability for clients to connect to the robot.

```
void setup()
{
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  pinMode(14,OUTPUT);
  pinMode(15,OUTPUT);
  pinMode(16,OUTPUT);
  WiFi.mode(WIFI_AP);
  WiFi.softAP("robotname");
  server.begin();
}
```

Controlling the robot

- Commands are needed to control the motion of the robot.
- A unique byte value is required to differentiate the motions.
- The table to the right shows the commands for the motions. A single letter will represent each motion.

Action	Command
Halt	S
Forward	F
Reverse	B
Left	L
Right	R

Receiving Commands

- In the loop() function, two things need to be checked.
  - Has a client connected to the robot?
  - Has a command been received?
- One big rule about writing code. No infinite loops in the loop() function.

loop() Function

- In the **loop()** function, the first thing that is checked is if a client is connected to the robot.
- The object **client** is assigned to a client that has connected. If no client has connected then the **client** object is empty or null.
- The **if()** statement checks if the **client** object is null or not. The result of the **if()** statement is always true if the variable is not empty or null.
- If a client has connected. the statement **Connected** will be displayed on the serial monitor.

```
loop() {
  client = server.available();
  if(client) {
    Serial.println("Connected");
  }
}
```

loop() Function

- A **while()** loop is created to process all commands while the client is connected. As long as the result of **client.connected()** is true, the code inside the **while()** loop will be executed.

```
loop() {
  client = server.available();
  if(client) {
    Serial.println("Connected");
    while(client.connected()) {
      while(!client.available()) {
        if(!client.connected()) break;
        delay(1);
      }
    }
  }
}
```

loop() Function

- The next highlighted line is where the code is looking for any commands sent to the robot. It works the same as **Serial.available()**.
- The **while()** loop here executes as long as there is no commands being sent. It does two things. First, it checks to make sure a client is still connected otherwise the **while()** loop will get stuck forever. Second, a **delay()** function is executed. This allows the processor to multi-task and handle WiFi operations.
- If the client disconnects, the **break** causes the code to exit the the **while()** loop.

```
loop() {
  client = server.available();
  if(client) {
    Serial.println("Connected");
    while(client.connected()) {
      while(!client.available()) {
        if(!client.connected()) break;
        delay(1);
      }
    }
  }
}
```

loop() Function

- After a command has been received, the code exits the while loop and then the command byte is read.
- Reading a byte from the client is the same as reading a byte from the serial interface.

```
loop() {
  client = server.available();
  if(client) {
    Serial.println("Connected");
    while(client.connected()) {
      while(!client.available()) {
        if(!client.connected()) break;
        delay(1);
      }
      char a = client.read();
    }
  }
}
```

loop() Function

- The command is then checked in the **switch()** statement.
- The **switch()** statement allows a variable to be tested against a list of values. If the variable matches the value to the right of the **case** statement, the code to the right of the colon is executed.
- A **break** statement is used to exit the **switch()** statement. If the **break** is not included, the processor would continue executing the rest of the code in the **switch()** statement.

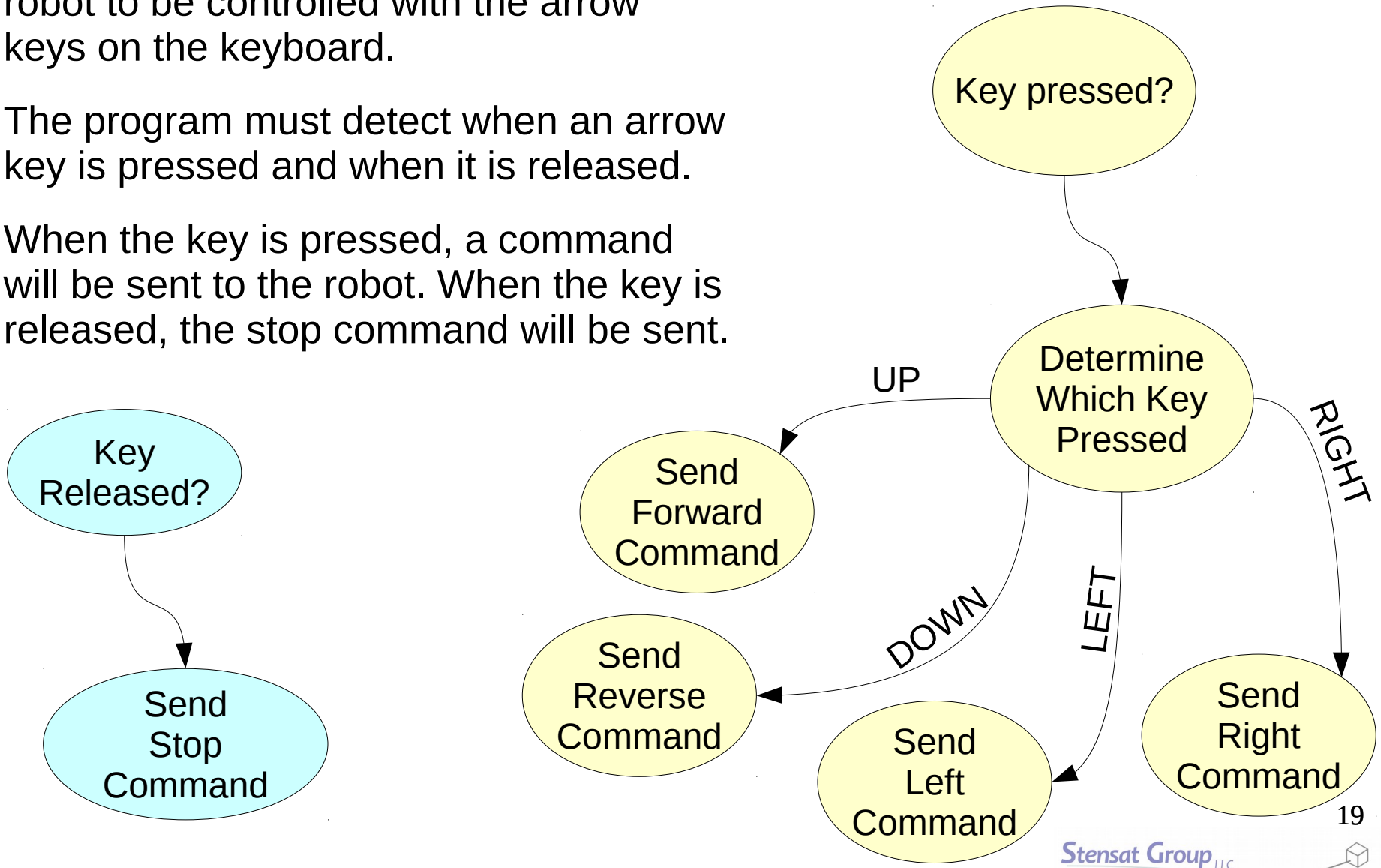
```
loop() {
  client = server.available();
  if(client) {
    Serial.println("Connected");
    while(client.connected()) {
      while(!client.available()) {
        if(!client.connected()) break;
        delay(1);
      }
      char a = client.read();
      switch(a) {
        case 'F' : forward();
                   break;
        case 'B' : reverse();
                   break;
        case 'L' : left();
                   break;
        case 'R' : right();
                   break;
        case 'S' : halt();
                   break;
    }
  }
}
```

Remote Control Software

- Upload the code to the robot.
- With the robot software completed, it is time to write a program on the laptop to provide the control.
- Processing software will be used.

Software Development Sequence

- The program to be written will allow the robot to be controlled with the arrow keys on the keyboard.
- The program must detect when an arrow key is pressed and when it is released.
- When the key is pressed, a command will be sent to the robot. When the key is released, the stop command will be sent.



Stensat Group LLC

Importing Libraries

- Start the Processing software.
- Click on the **Sketch** menu and select the **Import Library**
- Select the **network** library.
- The import command will appear at the top of the editor. This tells the compiler to include the library of functions to support network operations. This library will be used to access the robot over wifi.

```
import processing.net.*;
```

Setting up a TCP Port

- The next step is to create a network object that will provide access to the wifi.

```
import processing.net.*;

Client c;
```

Setting up a TCP Port

- In the setup() function, the network object is configured to talk to the robot at the IP address of 192.168.4.1 using port 80.
- A window of 800 by 600 pixels is also created.

```
import processing.net.*;

Client c;

void setup()
{
  size(800,600);
  c = new Client(this,”192.168.4.1”,80);
}
```

Detecting the Keys

- To detect the keys, two event functions will be used.
  - **keyPressed()** and **keyReleased()**
- **keyPressed()** is executed when a key on the keyboard is pressed.
- **keyCode** is a system defined variable that tells you what key was pressed.
  - The arrow keys have names in upper case to make it easier to use.

```
import processing.net.*;

Client c;

void setup()
{
  size(800,600);
  c = new Client(this,"192.168.4.1",80);
}

void keyPressed()
{
  if(keyCode == UP) c.write("F");
  else if(keyCode == LEFT) c.write("L");
  else if(keyCode == RIGHT) c.write("R");
  else if(keyCode == DOWN) c.write("B");
}
```


Stensat Group LLC

Detecting the Keys

- Notice that the network object **c** has a function called **write()**. This sends what is in quotes to the robot over wifi.
- For this to work, **Processing** requires the function **draw()** to exist even if there is no code in the function.

```
import processing.network.*;

Client c;

void setup()
{
  size(800,600);
  c = new Client(this,”192.168.4.1”,80);
}

void keyPressed()
{
  if(keyCode == UP) c.write(“F”);
  else if(keyCode == LEFT) c.write(“L”);
  else if(keyCode == RIGHT) c.write(“R”);
  else if(keyCode == DOWN) c.write(“B”);
}

void draw()
{
}
```

Detecting Key Release

- The code can now send commands to control the motion of the robot. What is missing is a way to stop the motion of the robot.
- The **`keyRelease()`** function is executed when a key is released.
- The only thing the function needs to do is send the stop command to the robot. A single **`c.write()`** function is used. The first argument has the **`S`** command. Remember the robot commands are terminated with a linefeed.

```
import processing.network.*;

Client c;

void setup()
{
  size(800,600);
  c = new Client(this,"192.168.4.1",80);
}

void keyPressed()
{
  if(keyCode == UP) c.write("F");
  else if(keyCode == LEFT) c.write("L");
  else if(keyCode == RIGHT) c.write("R");
  else if(keyCode == DOWN) c.write("B");
}

void keyReleased()
{
  c.write("S");
}

void draw()
{
}
```

Test Drive

- Now that the program is completed, it is time to test it.
- Turn on the robot.
- On the laptop, select the network icon to find available WiFi access points.
- Connect the laptop to the robot WiFi network.
- Start the program in Processing.
- Press the arrow keys and watch the robot move around.
- When the program starts, a window will open. This is the program running. If needed, click on the window to make it active.