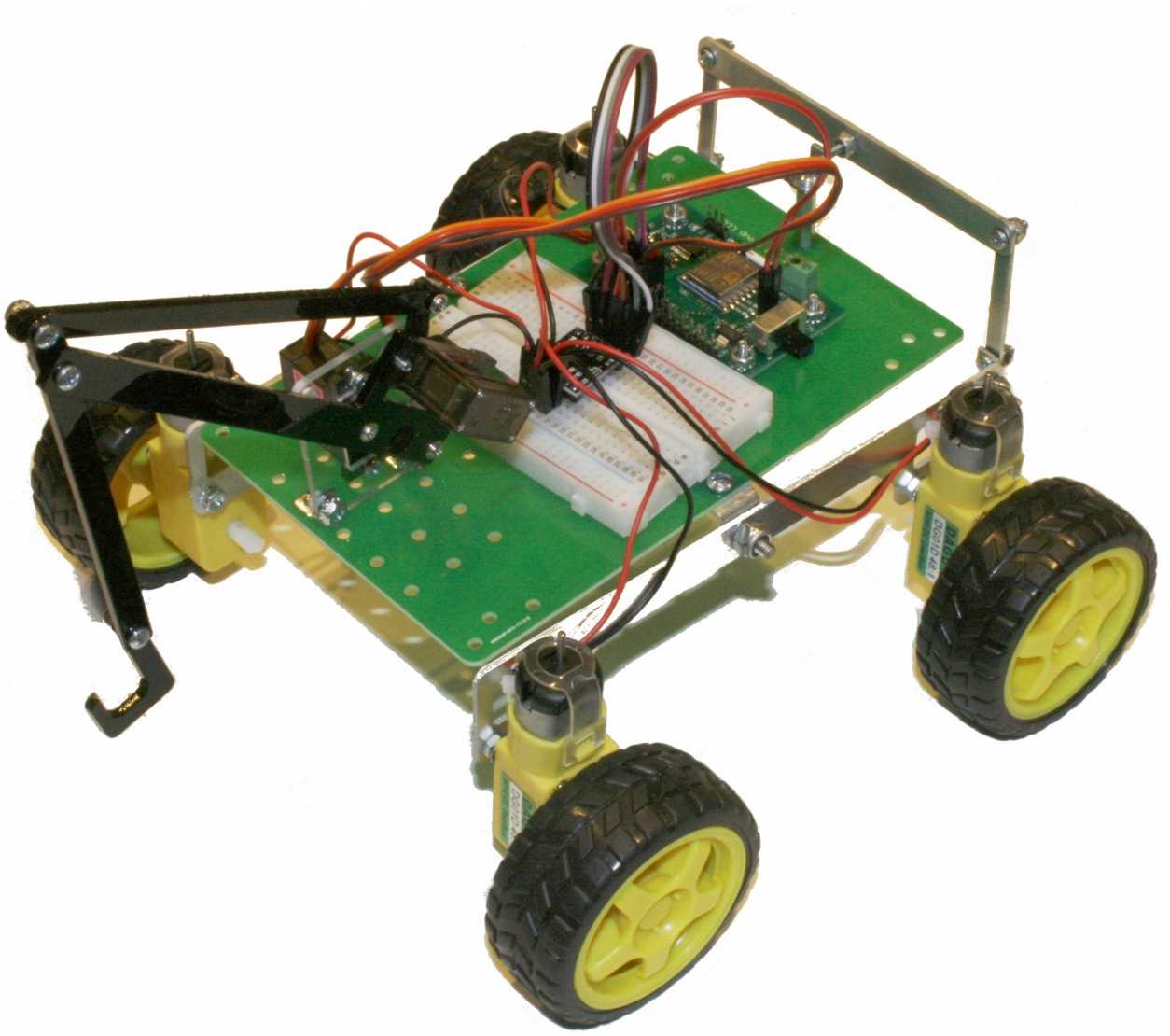
*Sten**BOT*** Robot Kit



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

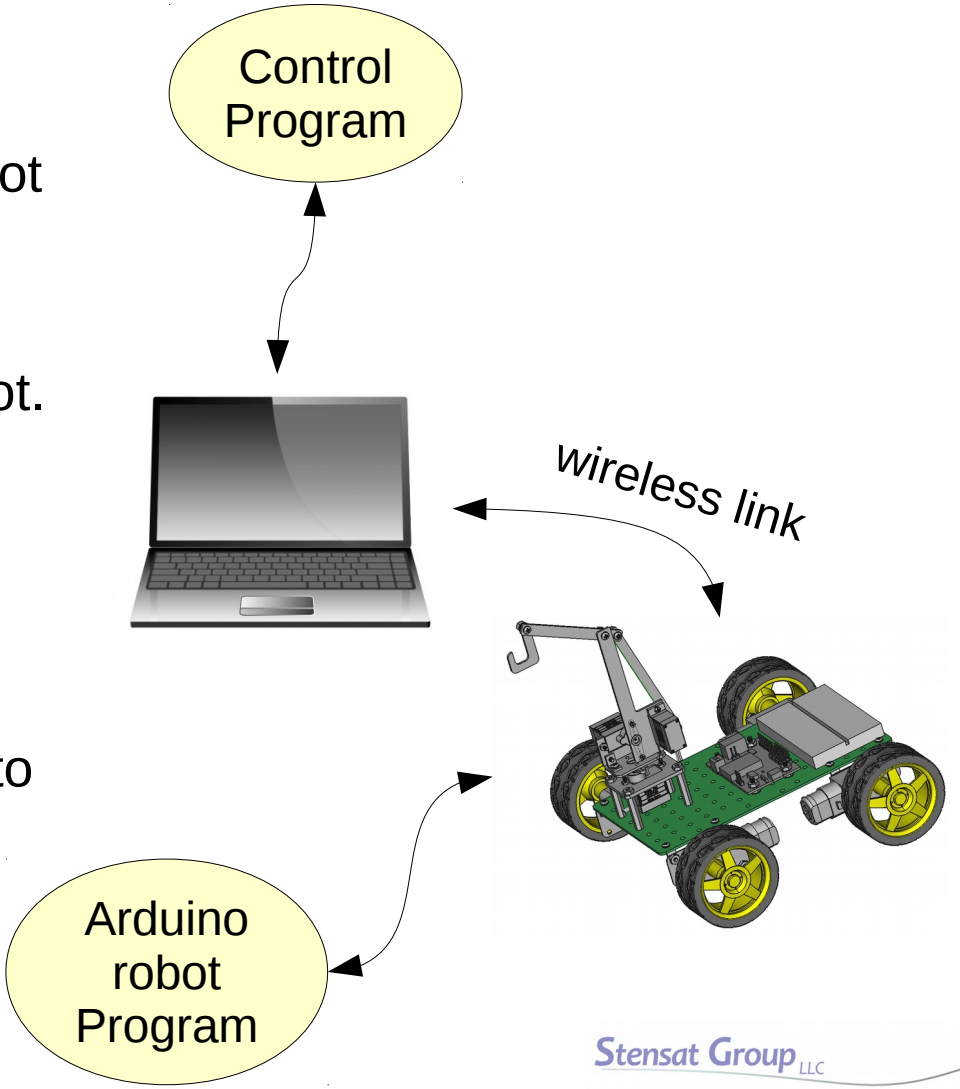
- www.arduino.cc
- http://esp8266.github.io/Arduino/versions/2.1.0/doc/reference.html

Introduction

- The wifi integrated in the processor provides a way to communicate with the robot kit. You can connect using a laptop.
- The robot will be configured as an access point. This means it becomes a local network where your laptop connects. It is also possible to have a tablet connect to the robot.
- In this lesson, you will learn how to control the robot and have the robot send data back. This will require you to write code on the laptop.

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 Wifi

- **WiFi** is a local area wireless computer network. It is also known as wireless local area network.
- **WiFI** is a standard for allowing computers to interact with each other using radio signals.
- A **wireless access point** is a device that connects a wireless network to a wired network. It can also provide a local isolated network not connected to the internet or other wired network. Access points usually have a network router and can provide network addresses or IP addresses to any device that connects.

Wifi Terms

- **SSID** – is a unique identifier for the WiFi network. It can have up to 32 characters and is case sensitive. This allows multiple WiFi access points in the same area without interfering with each other.
- **IP Address** – is the internet protocol address assigned to each device on the network. There are two standards, IP-4 and IP-6. IP-4 is used here. The address consists of four sets of numbers separated by a decimal point. Each number has a range of 0 to 255. Example 192.168.1.10.
- **DHCP** – is Dynamic Host Configuration Protocol. This protocol allows a WiFi router to assign an IP address to any device that connects to the WiFi network. This is done automatically.
- **TCP** – is Transmission Control Protocol. This is one of the main network protocols used by any device on any WiFi network or the internet. The protocol enables two devices to establish a connection to each other and exchange data. The protocol guarantees delivery of data and that the data is delivered in the same order sent.
- **UDP** – is User Datagram Protocol. This does not require two devices to connect to each other or verify delivery of data has been made. The sending device sends the packet to the receiving device and if it makes it, the data is received. It is possible for data to be lost.

TCP vs UDP

- TCP guarantees that data is delivered or if not, will provide an error indicating data was not delivered. TCP requires one device to establish a connection to another. When one device sends a packet to the other, the receiving device will respond to the sender that it has been received properly or not. If the receiver is not responsive, the sender will eventually quit trying and generate an error. TCP is a little slower in that time is spent with the sender waiting for the receiver to respond. TCP is used when data cannot be lost such as file transfers.
- UDP is faster in that there is no connection made between two devices. The receiver just listens for packets and takes them when received. The sender can send many packets quickly because there is no time waiting for the receiver to respond. The problem is that packets can get lost, either a bad connection or the receiver was not ready to receive a new packet. UDP can work for certain types of data such as video streams where an occasional loss is not catastrophic

WIFI Operation

- There are two parts to the WIFI operation
  - Configuration which sets up the module to operate properly
  - Data operation where the module receives data and can send data
- The WIFI module will be configured to operate as an access point. This allows another computer to connect to the module and communicate with the module.
- More than one WIFI access point can be in the same area and operate independent of each other as long as their SSID are different.
- In this lesson, the WIFI module will be configured as an access point and allow TCP connections.

Wifi Configuration

- First thing to do is include the **ESP8266WiFi** library by adding the three include statements to the top of the program.
- 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>
#include <WiFiServer.h>
#include <WiFiClient.h>


WiFiClient client;
WiFiServer server(80);

unsigned char cmd[6];
```

Wifi Configuration

- 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.
- Last operation is to set up the WiFi 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");
  delay(1000);
  server.begin();
}
```

Wifi Configuration

- Add the loop() function to the program if it isn't already included.
- Upload the program and let it run.
- On your computer, look up the available wireless networks and see if the one you named appears on the list. It may take a little while since the laptop OS checks for available networks at some interval of seconds.
- If it appears, try connecting. If you include a password, you should be prompted to enter a password.

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

void loop()
{
}
```

Programming 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() function

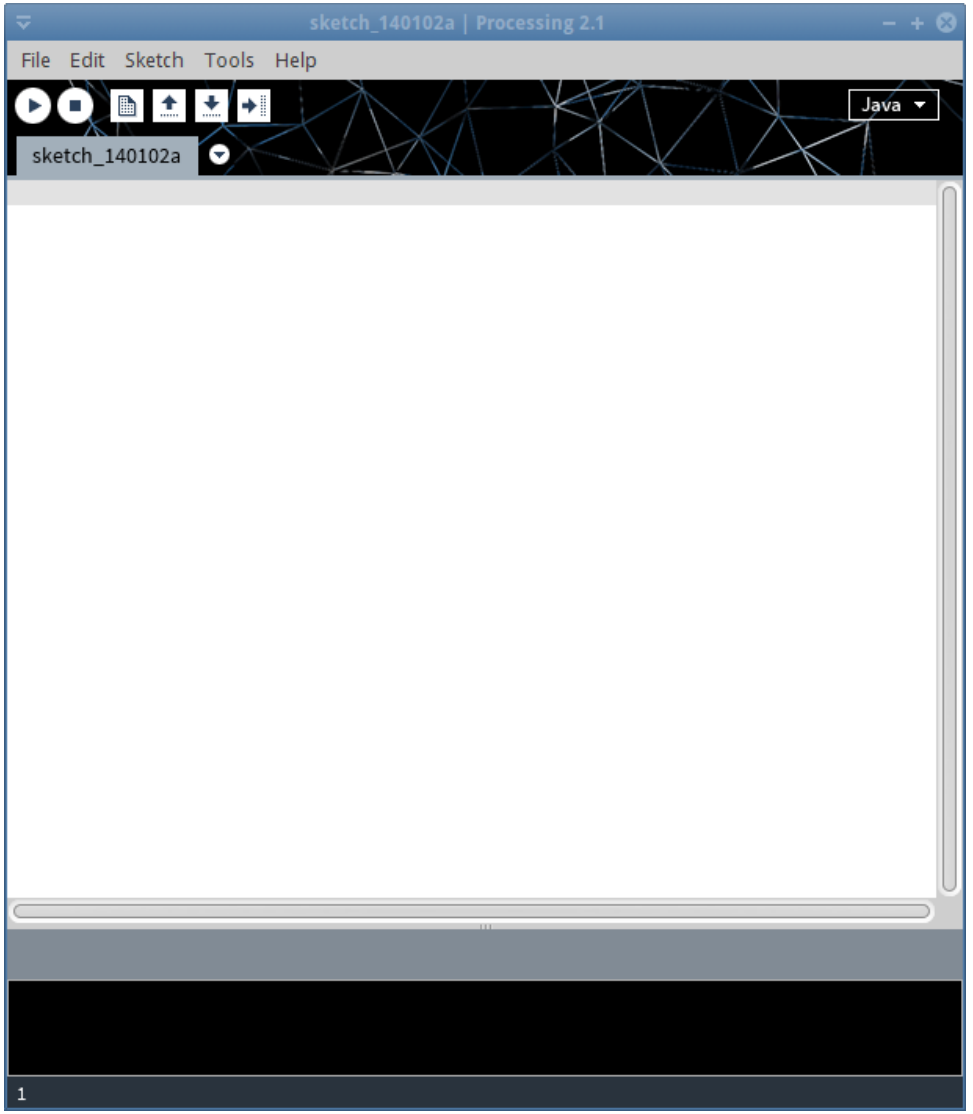
```
loop() {
  client = server.available();
  if(client) {
    Serial.println("Connected");
    while(client.connected()) {
      while(!client.available()) {
        if(!client.connected()) break;
        delay(1);
      }
      char a = client.read();
      switch(a) {
        case 'F' : forward();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

Remote Control Software

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

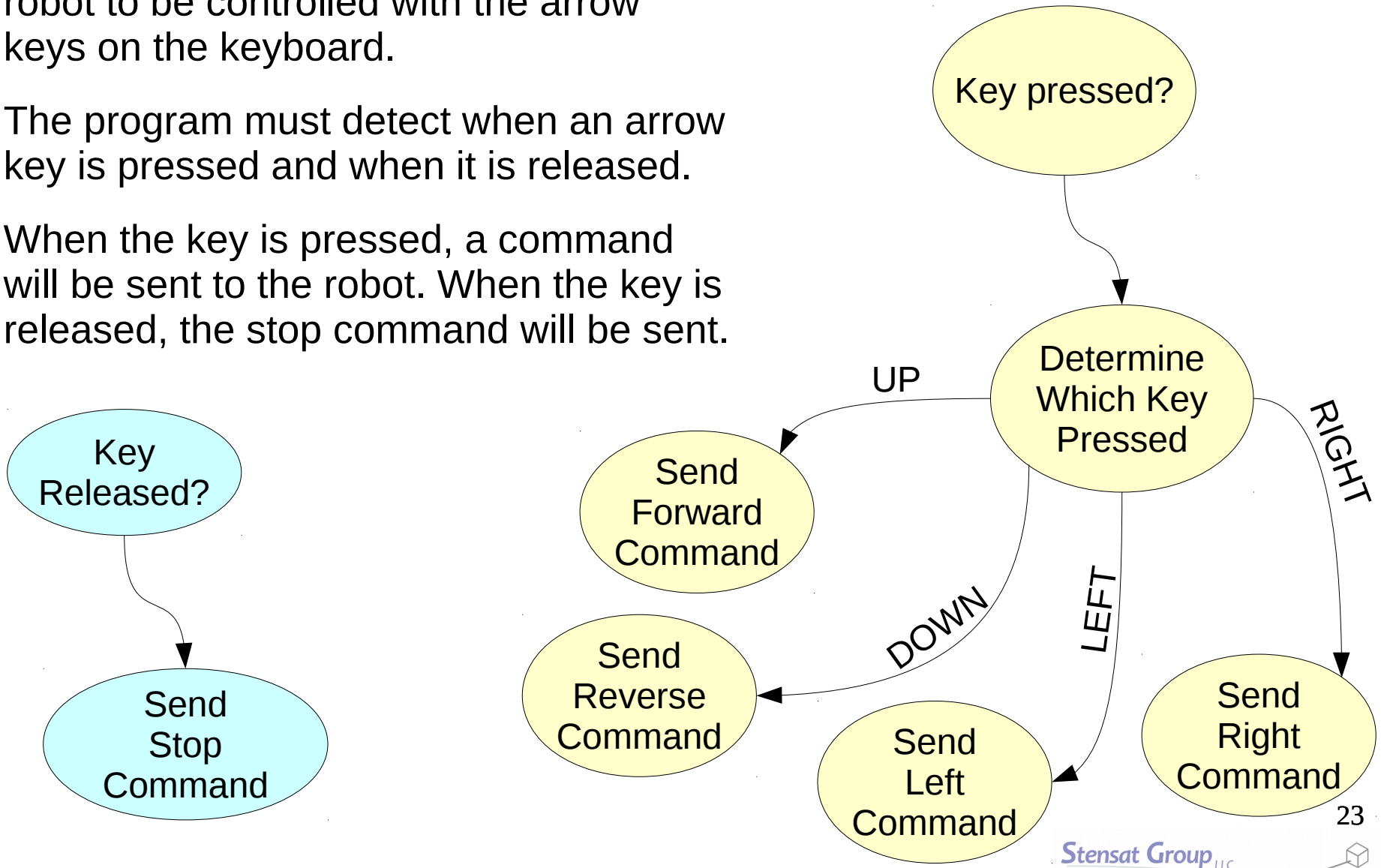
Software Development

- The development software to be used is called Processing.
- Processing Is a programming language and development environment using Java.
- Processing makes developing software easier.
- More information can be found at processing.org. The software is free to download and use.
- The development environment is similar to the Arduino development environment.



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.



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.network.*;
```

Setting up a TCP Port

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

```
import processing.network.*;

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.network.*;

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.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”);
}
```

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 with the wifi module connected.
- 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.
- If at first the laptop doesn't connect properly to the robot WiFi access point, cycle power to the robot and wait 30 seconds for the WiFi to start up and try connecting again. Some times the WiFi module does not quite work the first time it is programmed.

UDP Rover Control

- Both programs will be modified to handle UDP.
- In Processing, go to the **Sketch** menu and select **Add Library...**
- A window will open up. Scroll down toward the bottom. Locate the library called UDP and select it. At the bottom, select Install. The library will install.
- Once installation in complete, close the window.
- An internet connection is required to install the library.

UDP Code

- In the processing code, delete the import statement.
- Under the **Sketch** menu, select the library **UDP** and replace the existing import statement.
- Next, change Client to UDP.

```
import hypermedia.net.*;

UDP c;

void setup()
{
  size(800,600);
  c = new UDP(this,6000);
}

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()
{
}
```

UDP Code

- A string is created and set to the rover IP address.
- Change all the write() to send() as shown in bold.
- Since UDP does not make connections, the send function requires the destination address and port number all the time.

```
import hypermedia.net.*;

UDP c;
String ip = “192.168.4.1”;
void setup()
{
  size(800,600);
  c = new UDP(this,6000);
}

void keyPressed()
{
  if(keyCode == UP) c.send(“F”,ip,80);
  else if(keyCode == LEFT) c.send(“L”,ip,80);
  else if(keyCode == RIGHT) c.send(“R”,ip,80);
  else if(keyCode == DOWN) c.send(“B”,ip,80);
}

void keyReleased()
{
  c.send(“S”,ip,80);
}

void draw()
{
}
```

Rover Code Change

- The rover WiFi code needs to be modified to support UDP.
- Delete the everything above the **setup()** function. Enter the highlighted text above **setup()**.
- In the **setup()** function, remove **server.begin()** and replace it with **upd.begin()**.
- This sets up the processor to listen for packets received on the socket port specified by **localudpport** variable.

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

WiFiUDP udp;
unsigned int localudpport = 80;
char packet[255];

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

Rover Code Change

- In the loop() function, all the code to check for connection and available bytes gets deleted.
- The new first line checks if a packet has been received. It returns the number of bytes received or zero for no packets received.
- Next, the packet is read into the packet array. Up to 255 bytes will be read in. Only one byte is expected.
- If the read was successful, the first element of the packet is then used to determine which command was sent.

```
loop() {
  int packetsize = upd.parsePacket();
  if(packetsize) {
    int len = udp.read(packet,255);
    if(len > 0) {
      switch(packet[0]) {
        case 'F' : forward();
                   break;
        case 'B' : reverse();
                   break;
        case 'L' : left();
                   break;
        case 'R' : right();
                   break;
        case 'S' : halt();
                   break;
      }
    }
  }
}
```

UDP

- Upload the Arduino code into the SLATE.
- Run the Processing program and test it out.