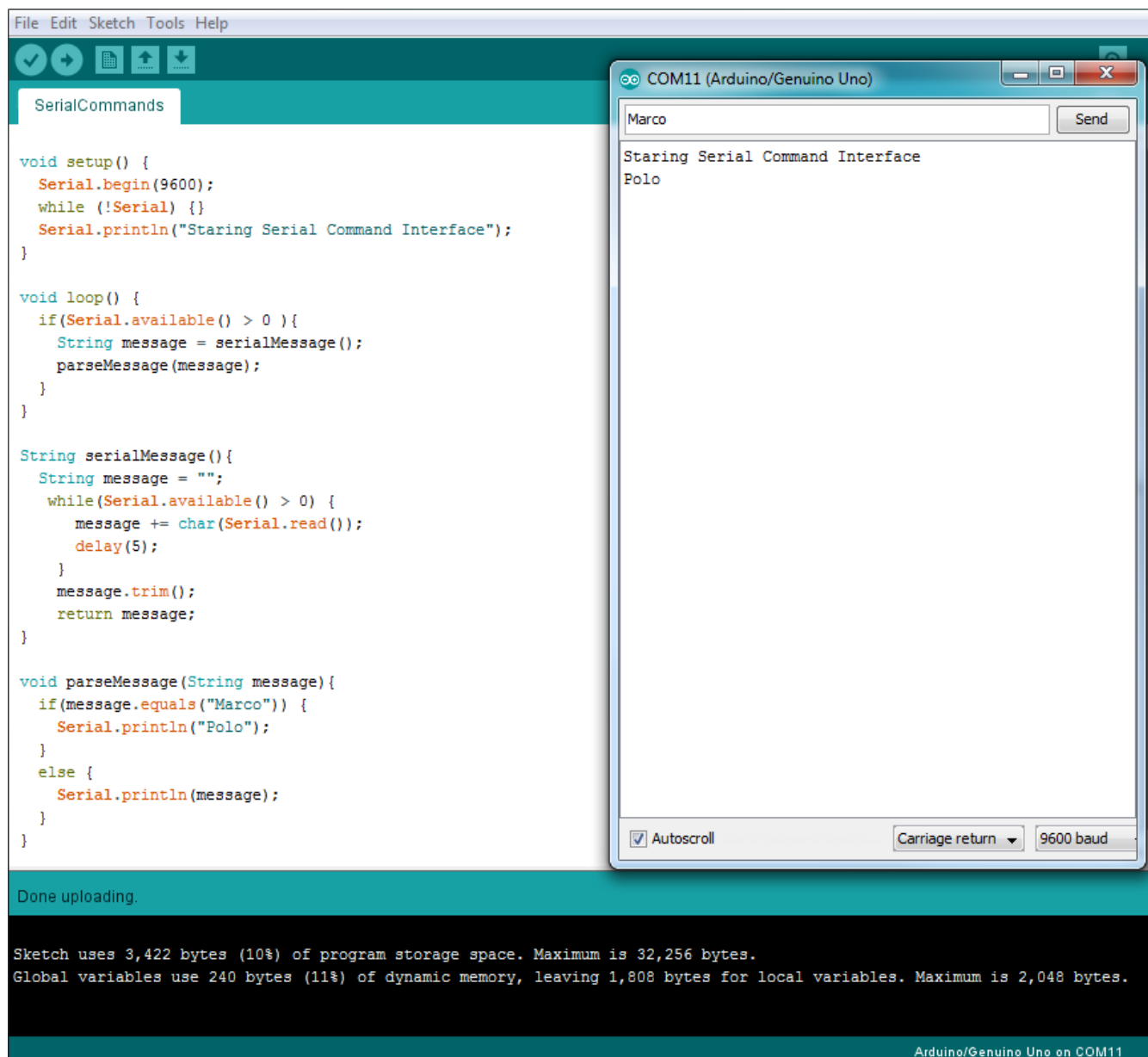


Serial Command Interface

Introduction

Open the included code file 'SerialCommunications.ino'. This sketch is designed to listen to any messages written to the serial line from the Arduino Serial Monitor. If that message is not a special predesignated command, the message is simply printed back out. If the message is a predesignated command, some special code is executed instead. In the current form, if you write the message 'Marco' it replies back "Polo."



The screenshot displays the Arduino IDE interface. The main window shows the 'SerialCommands' sketch with the following code:

```
void setup() {  
  Serial.begin(9600);  
  while (!Serial) {}  
  Serial.println("Staring Serial Command Interface");  
}  
  
void loop() {  
  if(Serial.available() > 0 ){  
    String message = serialMessage();  
    parseMessage(message);  
  }  
}  
  
String serialMessage(){  
  String message = "";  
  while(Serial.available() > 0) {  
    message += char(Serial.read());  
    delay(5);  
  }  
  message.trim();  
  return message;  
}  
  
void parseMessage(String message){  
  if(message.equals("Marco")) {  
    Serial.println("Polo");  
  }  
  else {  
    Serial.println(message);  
  }  
}
```

Overlaid on the right is the 'Serial Monitor' window for 'COM11 (Arduino/Genuino Uno)'. It shows the text 'Staring Serial Command Interface' and 'Polo' (a typo for 'Polo' in the original image). The input field contains 'Marco' and the 'Send' button is visible. At the bottom of the monitor window, 'Autoscroll' is checked, 'Carriage return' is selected, and the baud rate is '9600 baud'.

Below the code editor, a status bar indicates 'Done uploading.' and provides memory usage information:

Sketch uses 3,422 bytes (10%) of program storage space. Maximum is 32,256 bytes.
Global variables use 240 bytes (11%) of dynamic memory, leaving 1,808 bytes for local variables. Maximum is 2,048 bytes.

The bottom right corner of the IDE shows 'Arduino/Genuino Uno on COM11'.

Code Walkthrough

```
void setup() {  
    Serial.begin(9600);  
    while (!Serial) {}  
    Serial.println("Starting Serial Command Interface");  
}
```

The setup function starts by initializing a Serial connection at 9600 baud. Next, it waits until the Serial port is ready. Finally, it prints a message to be displayed by the Serial Monitor showing the device has started. This is almost identical to the behavior of the Repeater from 1.4.

```
void loop() {  
  
    if (Serial.available() > 0) {  
        String message = serialMessage();  
        parseMessage(message);  
    }  
}
```

The loop function continually checks the serial buffer using Serial.available to see if how many characters are in the buffer. If that number is more than zero, it calls the serialMessage() function and assigns the string it returns to a variable called message. It passes the message along to a function called parseMessage.

```
void parseMessage(String message) {  
  
    if (message.equals("Marco")) {  
        Serial.println("Polo");  
    }  
  
    else {  
        Serial.println(message);  
    }  
}
```

This function looks at the message we pass in (from the serial buffer) and checks to see if it matches the string 'Marco'. If it does, we print the string 'Polo.' If it does not, we simply echo back the original string.

Programming Challenge

using another if statement, extend the parseMessage word to return the string "Pong" if you the message is "Ping."