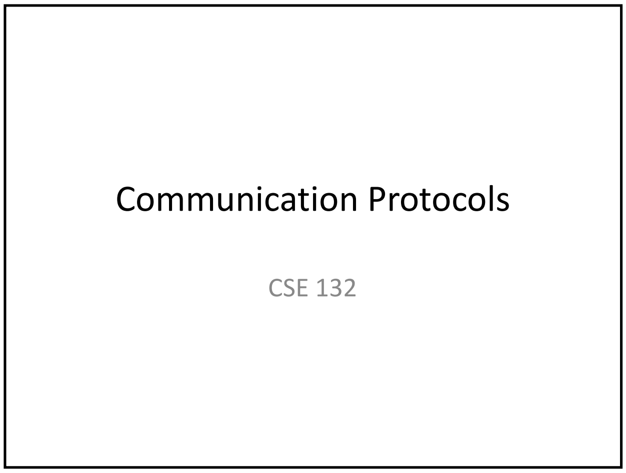
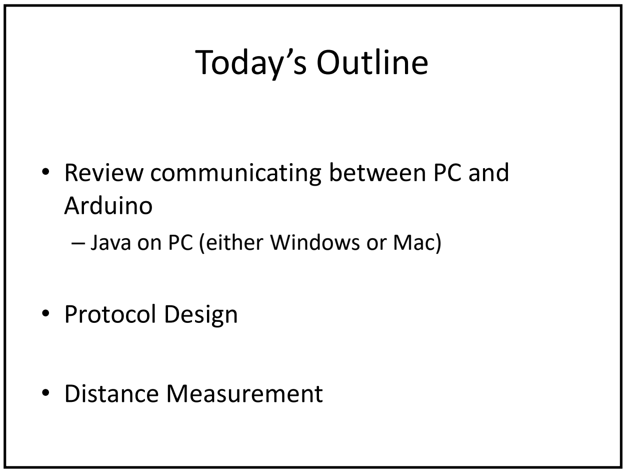


Communication Protocols

CSE 132

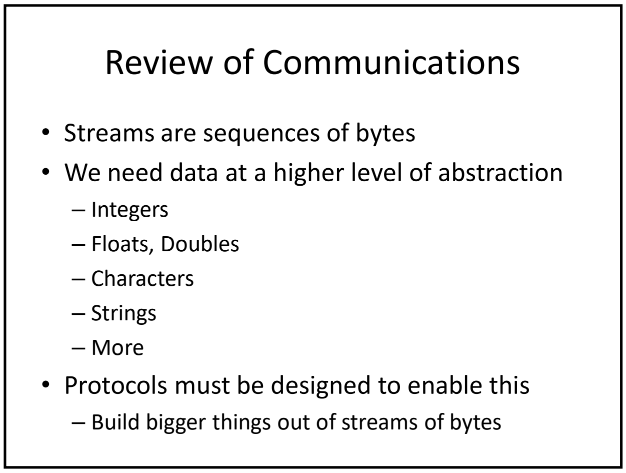
1



Today’s Outline

- Review communicating between PC and Arduino
  - Java on PC (either Windows or Mac)
- Protocol Design
- Distance Measurement

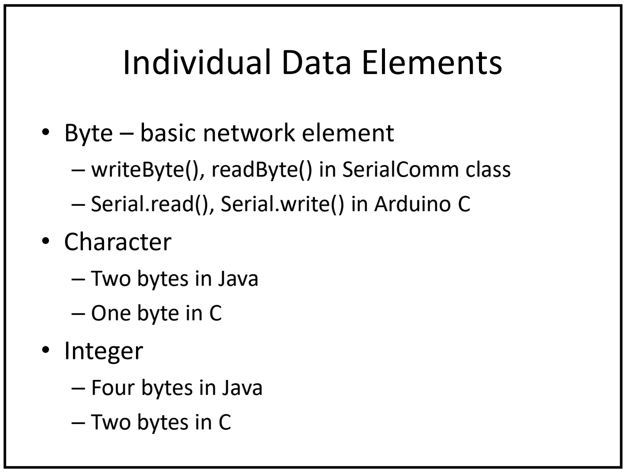
2



Review of Communications

- Streams are sequences of bytes
- We need data at a higher level of abstraction
  - Integers
  - Floats, Doubles
  - Characters
  - Strings
  - More
- Protocols must be designed to enable this
  - Build bigger things out of streams of bytes

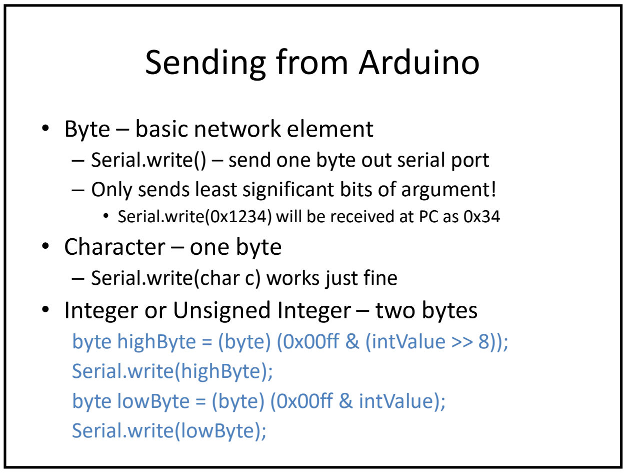
3



Individual Data Elements

- Byte – basic network element
  - writeByte(), readByte() in SerialComm class
  - Serial.read(), Serial.write() in Arduino C
- Character
  - Two bytes in Java
  - One byte in C
- Integer
  - Four bytes in Java
  - Two bytes in C

4

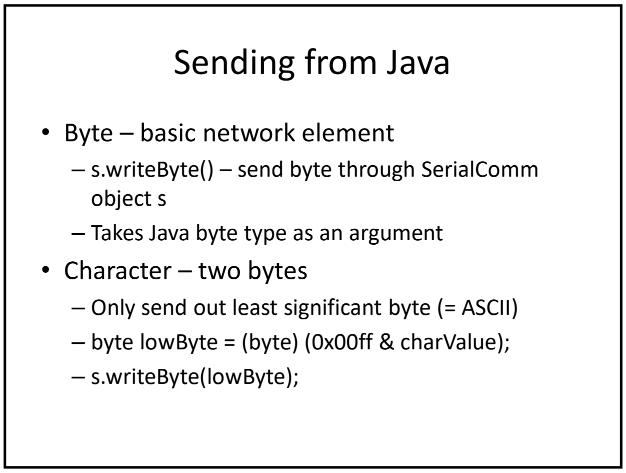


Sending from Arduino

- Byte – basic network element
  - Serial.write() – send one byte out serial port
  - Only sends least significant bits of argument!
    - Serial.write(0x1234) will be received at PC as 0x34
- Character – one byte
  - Serial.write(char c) works just fine
- Integer or Unsigned Integer – two bytes

```
byte highByte = (byte) (0x00ff & (intValue >> 8));
Serial.write(highByte);
byte lowByte = (byte) (0x00ff & intValue);
Serial.write(lowByte);
```

5



Sending from Java

- Byte – basic network element
  - s.writeByte() – send byte through SerialComm object s
  - Takes Java byte type as an argument
- Character – two bytes
  - Only send out least significant byte (= ASCII)
  - byte lowByte = (byte) (0x00ff & charValue);
  - s.writeByte(lowByte);

6

Sending from Java

- Integer – 4 bytes in Java

```
byte b1 = (byte) ((intValue >> 24) && 0xff);
s.writeByte(b1);
byte b2 = (byte) ((intValue >> 16) && 0xff);
s.writeByte(b2);
byte b3 = (byte) ((intValue >> 8) && 0xff);
s.writeByte(b1);
byte b4 = (byte) (intValue && 0xff);
s.writeByte(b2);
```

- But what if receiver is only expecting 2 bytes?

7

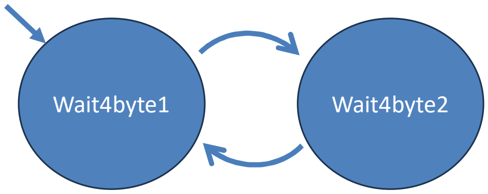
Receiving

- First check to see if byte has arrived
  - Arduino
    - Serial.available() returns integer count of available bytes
  - Java
    - s.available() returns Boolean if a byte is available
- Next read one (and only one) byte
  - Serial.read() on Arduino (returns an int)
    - Return value is -1 if nothing received, byte value otherwise
  - s.readByte() on Java (returns a byte)
- Check available() prior to *each* read!

8

FSM to Receive 2 Byte Integer

- Initial state: Wait4byte1



- State transition: receipt of a byte
  - available() followed by read()
  - Save incoming byte on transition

9

Receive 2 Byte Integer

- Use FSM to save b1 (first byte) and b2 (second)
- Compose int from 2 bytes:

```
int value = (b1 << 8) + b2;
```

- Watch out for sign extension in Java

```
int value = ((0xff & b1) << 8) + (0xff & b2);
```

  - Because bytes get promoted to int before math operations (silly Java rule)

10

Communications and Delta Time

```
while (true)
        now = millis()
        if (byte available) then
                read byte and save (i.e., FSM)
        endif
        if (delta time has expired)
                do time-based stuff
        endif
endwhile
```

11

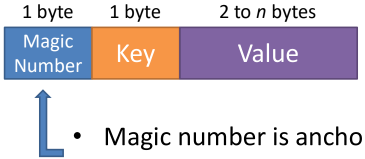
Protocol Design

- Communicating more than just a 2-byte int
- What do we want to communicate?
- How do we want to say it?

12

A Protocol for Us

Message format:



1 byte | 1 byte | 2 to *n* bytes

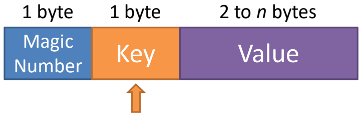
Magic Number | Key | Value

- Magic number is anchor of message
- Always first byte
- Unlikely to be in rest of message
- Reader can ignore bytes until it sees magic number and then receive

13

A Protocol for Us

Message format:



1 byte | 1 byte | 2 to *n* bytes

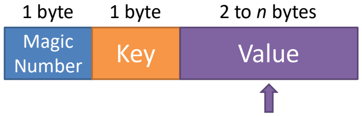
Magic Number | Key | Value

- Key tells what type of message
- Indicates both size and interpretation
- E.g., 2-byte temperature value
- E.g., 4-byte timestamp
- E.g., UTF-8 encoded error string
- Table of legal keys must be maintained

14

A Protocol for Us

Message format:



1 byte | 1 byte | 2 to *n* bytes

Magic Number | Key | Value

- Actual content of message
- Key tells how to interpret

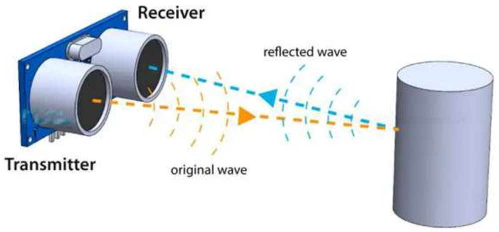
15

UTF-8 Strings

- New string format – different than Java or C
- Used for communication between machines
- Composition
  - 2-byte length (msb followed by lsb)
  - “length” number of UTF-8 characters (each 1 byte)
  - Maximum length of 100 bytes (non-standard extension for our class)
- Design FSM to receive UTF-8 strings
- Actually, use FSM to receive msgs in protocol

16

Ultrasonic Ranging



- Measure time for ultrasonic pulse to travel to target and back
- Translate time to distance using speed of sound
- Divide by 2, because sound traveled distance to target twice

17

Upcoming Schedule

- Assignment 5 due Monday, Mar 3
- Studio 6 is Monday, Mar 3
  - Use ultrasonic range finder
- Assignment 6 is due Monday, Mar 17 (first day after break)
  - Requires use of protocol
  - Read chapter 12 for example FSM
- No office hours during break, no late tickets charged

18