

Communication Protocols

CSE 1302

Today's Outline

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

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

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

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);
```

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);`

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);
```

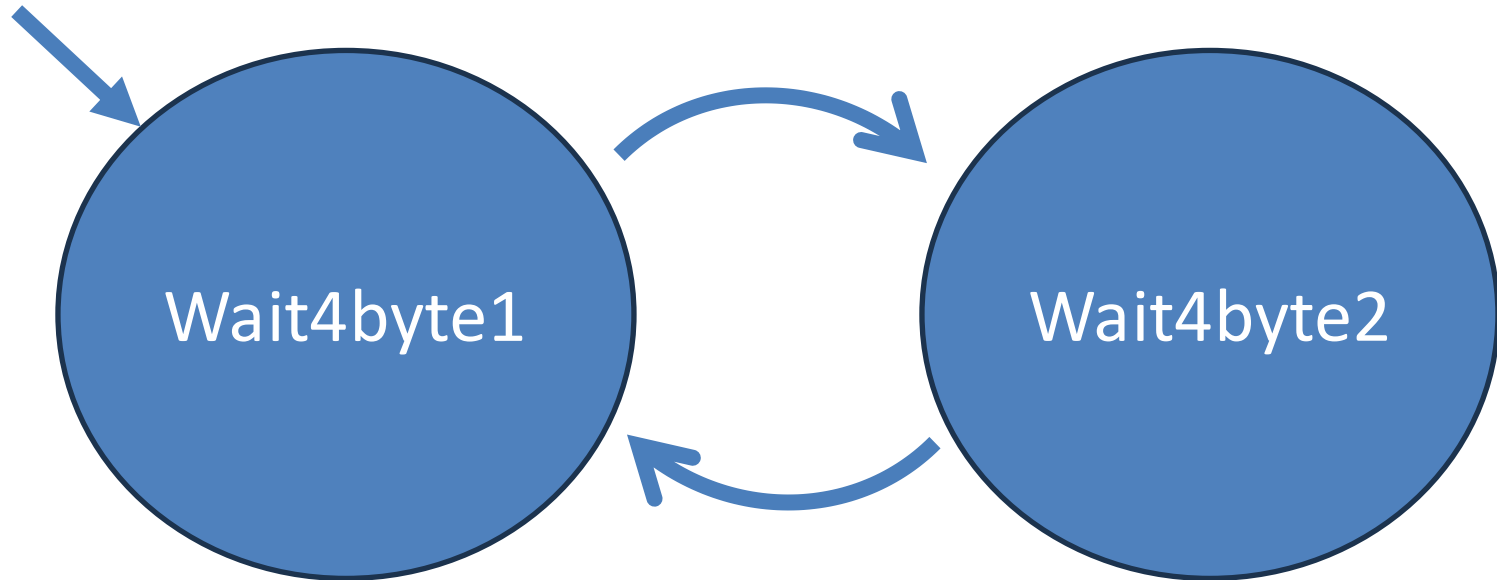
- But what if receiver is only expecting 2 bytes?

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!

FSM to Receive 2 Byte Integer

- Initial state: Wait4byte1



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

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)

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
```

Quiz Time

- Go to Canvas and answer the single question on Quiz 6A

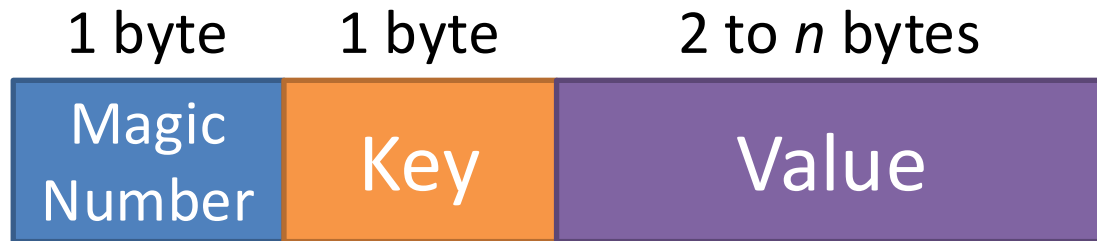
Serial.write() on the Arduino will write how many bytes to the serial stream when passed an int?

Protocol Design

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

A Protocol for Us

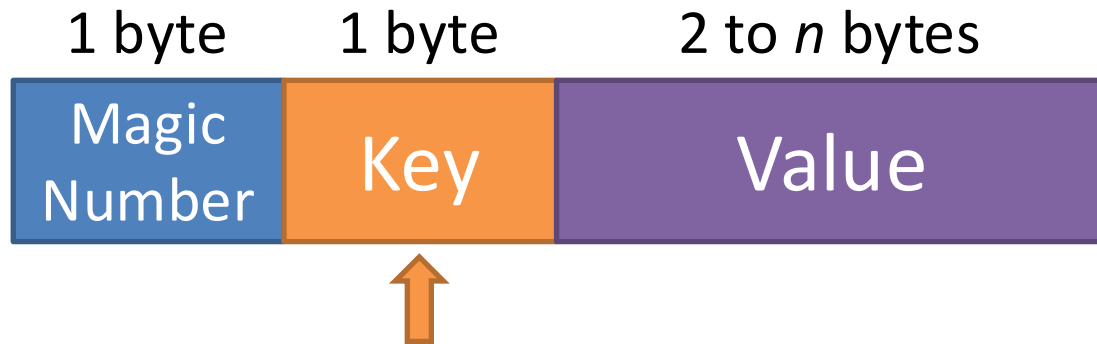
Message format:



- 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

A Protocol for Us

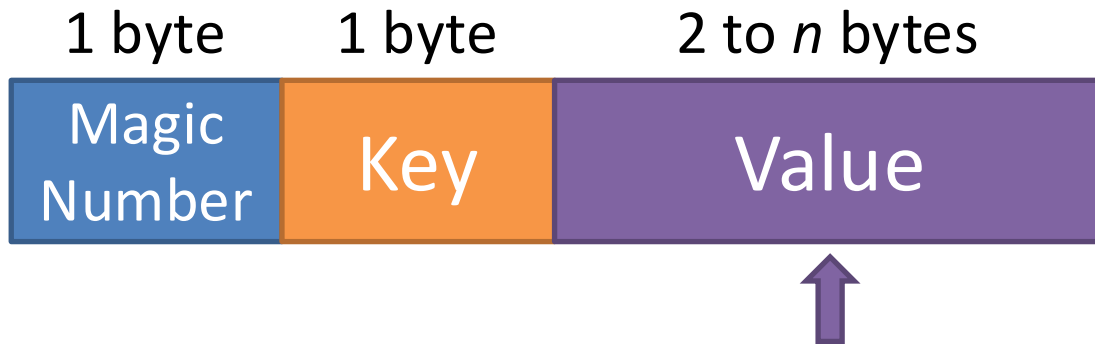
Message format:



- 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

A Protocol for Us

Message format:

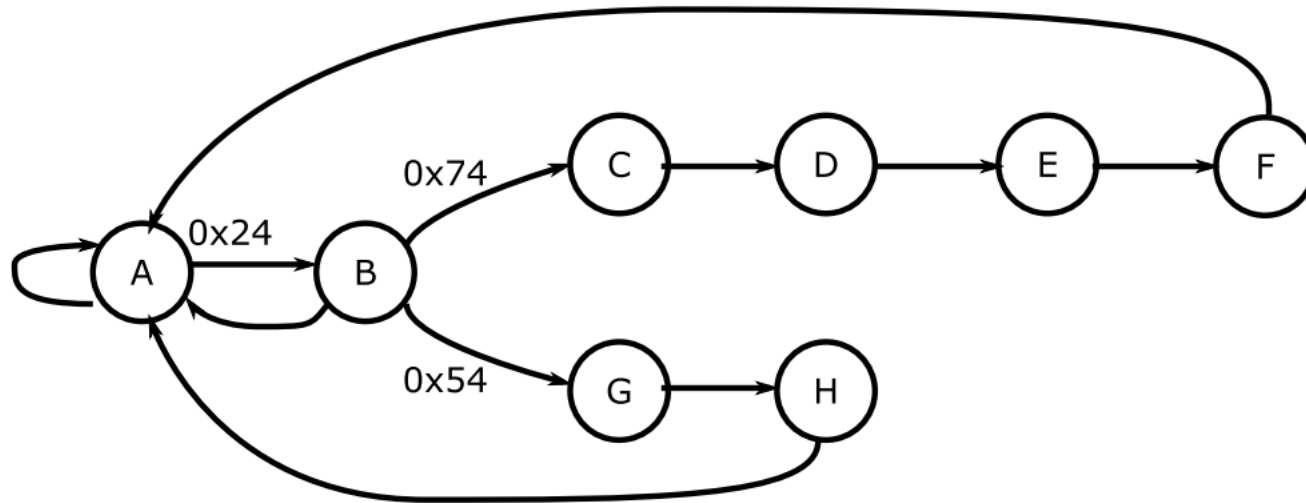


- Actual content of message
- Key tells how to interpret

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

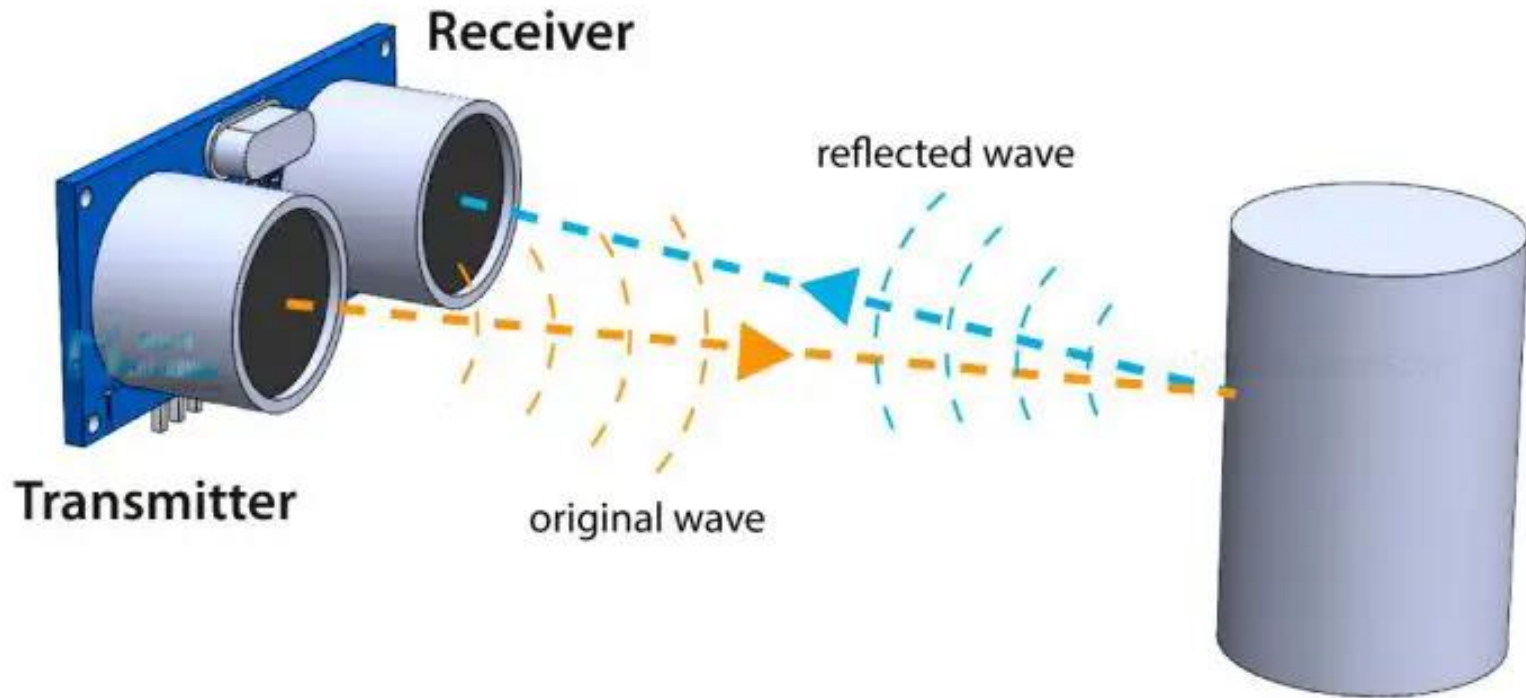
Example FSM of Our Protocol



State	Meaning
A	Waiting for magic number (initial state)
B	Waiting for key
C	Waiting for first byte of 0x74 msg value
D	Waiting for second byte of 0x74 msg value
E	Waiting for third byte of 0x74 msg value
F	Waiting for fourth byte of 0x74 msg value
G	Waiting for first byte of 0x54 msg value
H	Waiting for second byte of 0x54 msg value

Figure 12.1: Receiver finite-state machine.

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

Exam 1 Review

Exam is next Week!

Exam Logistics and Style

- Date and Time
 - Oct 8, here, Lopata 101
 - 1pm to 2:20pm, starting right at 1!
 - (or 2:30pm to 3:50pm, starting right at 2:30!)
- Questions
 - Question 1 will be a collection of short answer things (e.g., true/false, fill in the blank, quick definition)
 - Questions 2 through N will be longer (going more in depth on a particular subject)
- One-page “crib/cheat sheet” is allowed
 - 8.5 x 11 sheet, front and back, whatever you want to include (content-wise)

Practice and Help

- B quiz questions are mostly from old exams
- Extra credit opportunity
 - “Extra Credit Quiz 1E,” on Canvas, due before exam
 - Practice exam questions (from old exams)

Information Representation

- Number systems
 - Binary, two's complement, hex – conversions
 - Other negative representations
 - Fixed point – Q notation
 - Floating point – definitions
- Text representations
 - ASCII
 - UTF (-8, -16, -32)

Finite State Machines

- What are they? What are they good for?
- Bubble diagrams
 - Reading (i.e., what does this FSM do?)
 - Writing (e.g., author an FSM to do X)
- Implementation
 - Authoring in Arduino C
 - What constitutes/triggers a state transition?

Programming When Time Matters

- Simple delays
 - Advantages and disadvantages
 - How to program
- Delta time
 - When it really matters
 - Advantages and disadvantages
 - How to program

Input and Output

- Analog Input
 - Linear calibration, scaling, units, ranges
 - Simple filtering
 - Programming interface
- Digital Input/Output
 - Meaning, polarity
 - Programming interface
- Physical construction
 - Wiring
 - Components – resistors, LEDs, potentiometer, temperature sensor

Practicalities

- How to use development environment(s)
- Commonly used library functionality
 - Controlling pins (in and out)
 - Printing to attached PC
 - Timing
- Details of Arduino C language
 - Standard data types
 - Similarities and differences relative to Java
 - Bit-level and logical manipulation

Not Covered in Exam 1

- Communications topics
- This material will be on exam 2