Arduino

AS220 Workshop

Part IV – *Communication*

Lutz Hamel
hamel@cs.uri.edu
www.cs.uri.edu/~hamel/as220

AS220
LABS

SOUND
PIXEL GRIDS
MOVING PIXELS
FOLK ELECTRONICS

Communication

We need two things in order to communicate:

- Medium or Carrier
  - the *physical aspect* of the communication
- Protocol
  - the *format* of the communication

Wired Communication

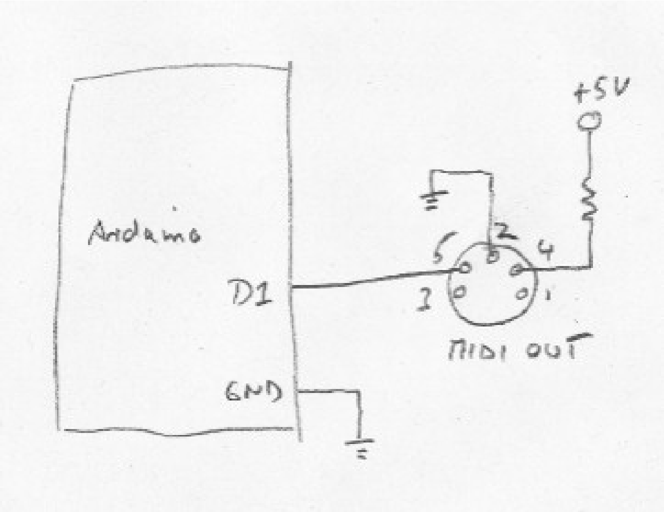
- RS232
  - point-to-point communication
  - specifies speed and format of each byte to be transmitted
    - e.g. 9600 bits/sec, 8 data bits, 1 stop bit, no parity
  - it is a very low level protocol, only specifies how to move bits from one computer to the next
  - no command structure

Wired Communication

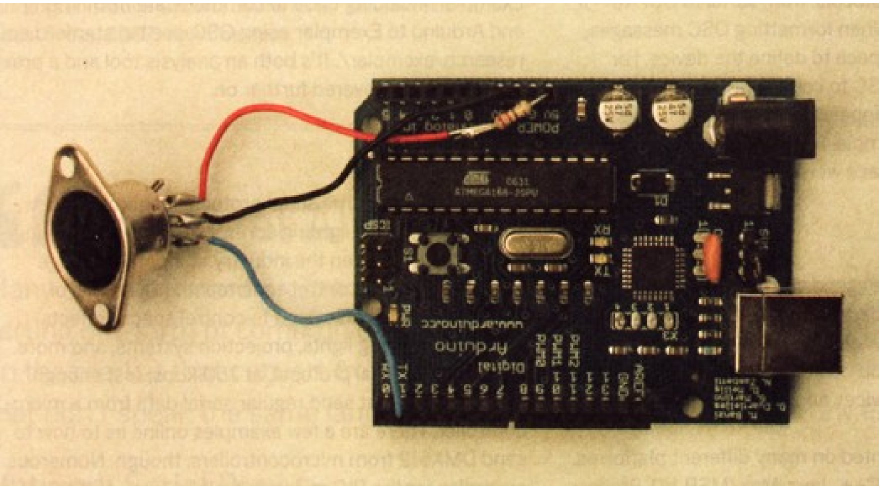
- MIDI (Musical Instrument Digital Interface)
  - allows synths, drum machines, *etc.* to talk to each other
  - uses RS232 at the lowest level but adds a three byte command structure
    - byte1: command (e.g. note on/off)
    - byte2: status (e.g. pitch)
    - byte3: status (e.g. velocity/touch intensity)
  - Can be daisy chained

Arduino

MIDI



```
/*
 * A simple MIDI program - continuously
 * play tone A at 440Hz on channel 1.
 */

void setup() {
  Serial.begin(31250); // MIDI baud rate
}

void loop() {
 // value 0x90 is channel 1
 // value 69 is A440
 // value 100 is medium velocity
 // value 0 is silent velocity
 noteon(0x90,69,100);
 noteon(0x90,69,0);
}

void noteon(char chan, char pitch, char vel) {
  Serial.print(chan, BYTE);
  Serial.print(pitch, BYTE);
  Serial.print(vel,BYTE);
}
```



NOTE: Unplug MIDI cable when uploading programs (D1 is the TX part of the serial communication to the Arduino IDE).

Source: “Making Things Talk”, Igoe, O'Reilly, 2007.

Network Protocols

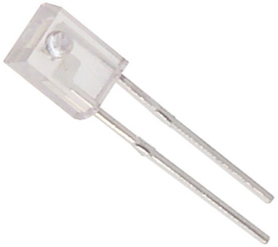
- Network protocols
  - are high level protocols that allow for general networking
    - TCP/IP (the internet protocol)
  - can use many different carriers
    - TCP/IP can run on wired and wireless carriers
  - are usually *packet oriented*
    - rather than packaging individual bytes they specify how to package larger chunks of data (e.g. 128 bytes at a time)

Wireless Communication

- In wireless communication we use an alternative carrier to carry our protocol
  - sound
    - sonar underwater communication
  - infrared (IR)
    - remote controls
  - radio frequency (RF)
    - wireless router

IR Remote Control

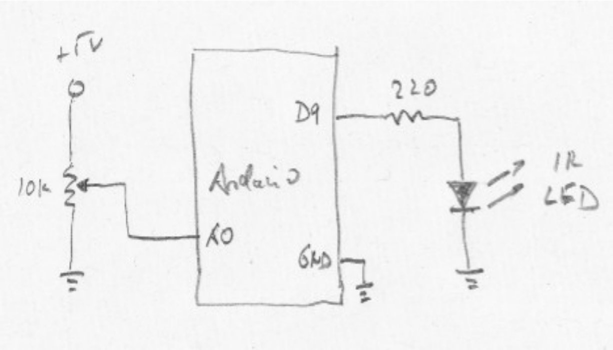
- Idea
  - We use two Arduino boards
    - transmitter using IR LED
    - receiver using IR phototransistor
  - We send an IR PWM signal from the transmitter to the receiver and the receiver will drive a DC motor according to the duty cycle of the PWM signal

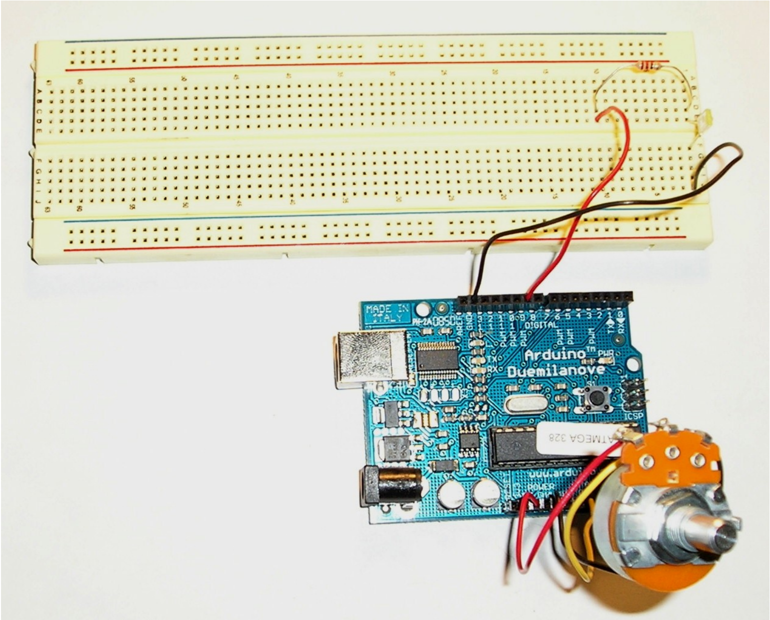


IR Remote Control

- Implementation Notes:
  - we use infrared light as our carrier
  - our protocol is PWM
    - on the transmitter side we modulate the carrier using PWM
    - on the receiver side we retrieve the PWM signal by sampling the received signal.
  - receiver side is tricky
    - typically a weak signal – amplification
    - we then sample the 500Hz PWM signal 4 times/msec and rebuild a PWM signal at the digital output pin for the motor.

IR Remote - Transmitter



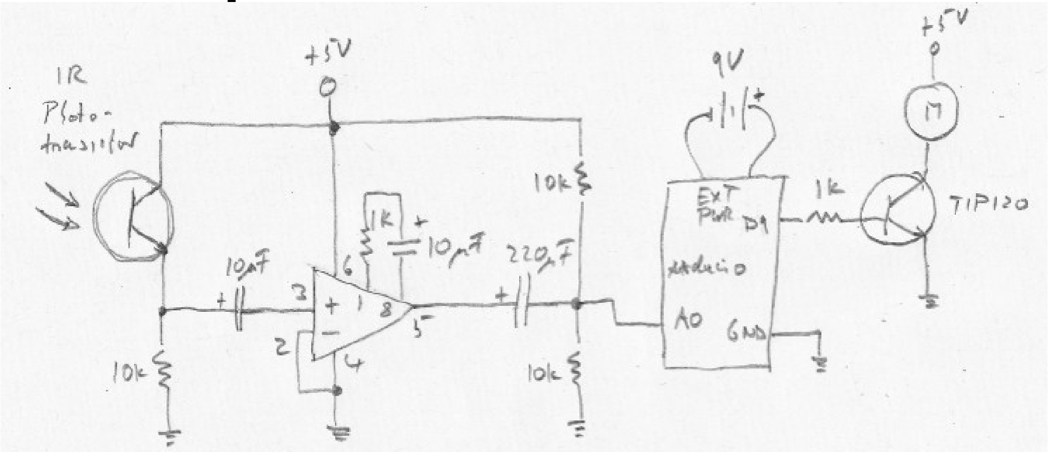


```
// Transmitter
// This is the transmitter for the IR remote control
// We read the pot value and send out a PWM signal on
// digital pin 9.

// pot connected to analog pin 0
int potPin = 0;
// IR LED connected to digital pin 9 (PWM)
int ledPin = 9;
// variable to store the value coming from the sensor
int val = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  val = analogRead(potPin);
  // we never want to generate DC so
  // adjust the range - 50 to 200
  // instead of 0-254
  val = map(val,0,1023,50,200);
  analogWrite(ledPin,val);
  delay(100);
}
```

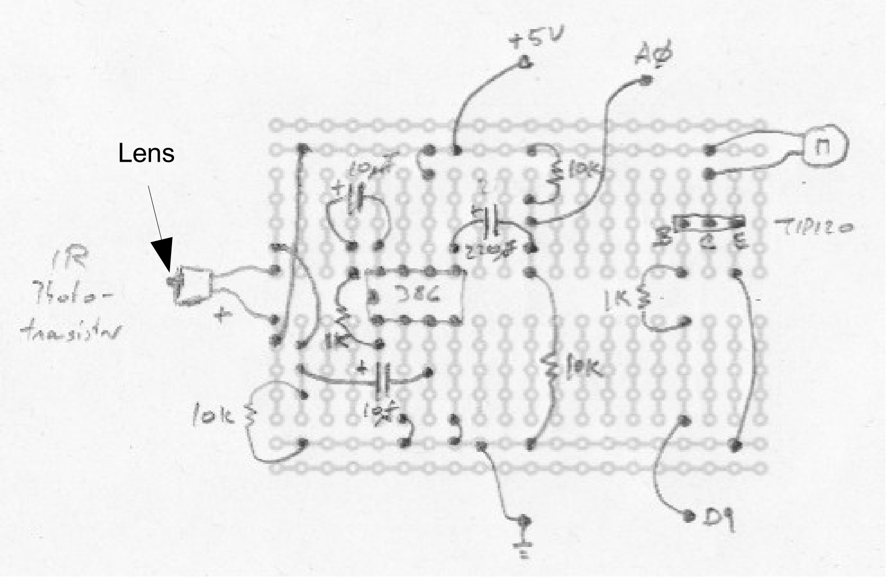
Note: The IR LED has a yellow top.

Arduino

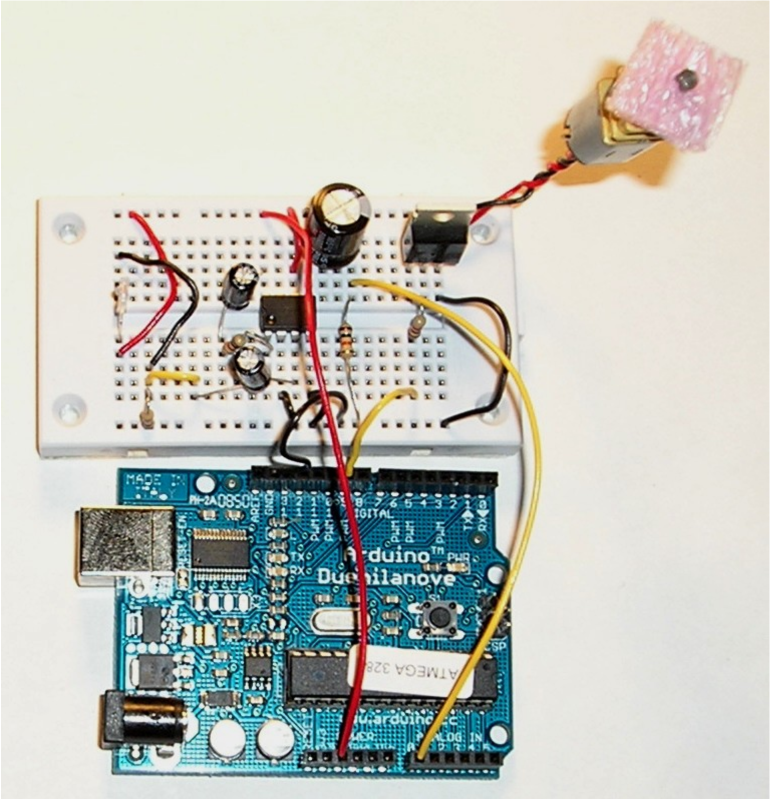
IR Remote - Receiver

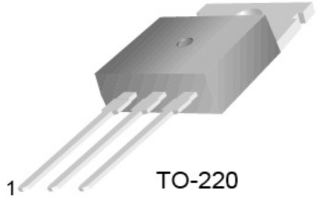


Note: The IR phototransistor has a red top.



Lens





1.Base 2.Collector 3.Emitter

IR Remote - Receiver

```
// Receiver
// This is the receiver for the IR remote control
// We read the analog signal value and send out a signal on
// digital pin 9. We sample the PWM signal coming in from
// the IR receiver and send out the appropriate hi/lo on the
// output pin in essence simulating the original PWM signal.

int signalPin = 0;   // signal connected to analog pin 0
int pwmPin = 9;      // motor connected to digital pin 9 (PWM)
int val = 0;         // variable to store the value coming from the sensor
int threshold = 650; // any value higher than this is considers HIGH

void setup() {
  pinMode(pwmPin, OUTPUT);  // declare the pwmPin as an OUTPUT
}

void loop() {
  val = analogRead(signalPin);
  if (val >= threshold) {
    digitalWrite(pwmPin, HIGH);
  }
  else {
    digitalWrite(pwmPin, LOW);
  }
  delayMicroseconds(250); // sample 4 x a millisecond
}
```

What to do Next

- Lots of interesting books to explore
  - “Practical Electronics for Inventors”, Scherz, McGraw-Hill, 2006.
  - “Physical Computing”, O'Sullivan and Igoe, Thomson, 2004.
  - “Making Things Talk”, Igoe, O'Reilly, 2007.
  - “Electronic Sensor Circuits & Projects”, Mims, Master Publishing, 2004.

Summary

- Basics
  - Blink, Reading Digital Input, Reading Analog Input, PWM and Dimming, Sound waves
- Interactive Design & Advanced Transducers
  - “The Loop”, Driving RC Servos, Driving DC Motors, Flexsensors, H-Bridge
- Multimedia Applications
  - Processing

Summary

- Communication
  - physical aspects, format, different protocols, wireless, Arduino IR remote control

Finally

- Go out there and build stuff!
- Most importantly: have fun!
- If you have questions give me a holler at:

hamel@cs.uri.edu