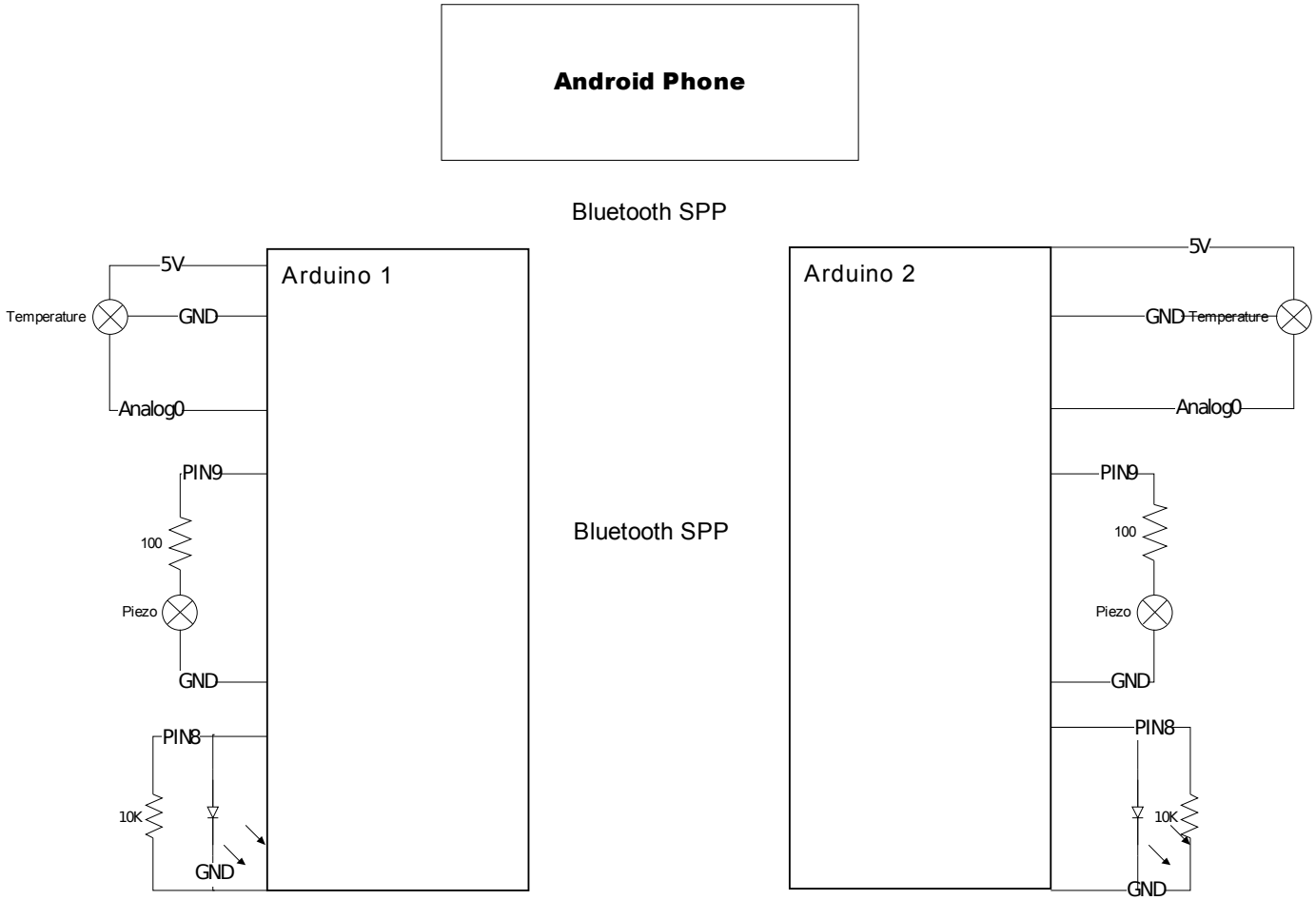
Bluetooth and Temperature Sensors

Ed Ruffing

COT4930 Dr. Shankar

11-30-10

Bluetooth Temperature



Bluetooth Support

- iWRAP API
- Signal temperature sensors from android phone using Bluetooth Chat client.
- Each Arduino signal each other when either detects a temperature threshold surpassed. Will require pairing of Arduino Bluetooth devices.

Temperature Readings

```
int calculateThresholdTemperature()
{
  int sensorTotal = 0;
  int sensorAverage = 0;
  for (int i = 0; i < 20; i++)
  {
    delay(200);
    sensorAverage = analogRead(sensorPin);
    sensorTotal = sensorTotal + sensorAverage;
  }
  sensorAverage = sensorTotal / 20;
  return sensorAverage;
}
```

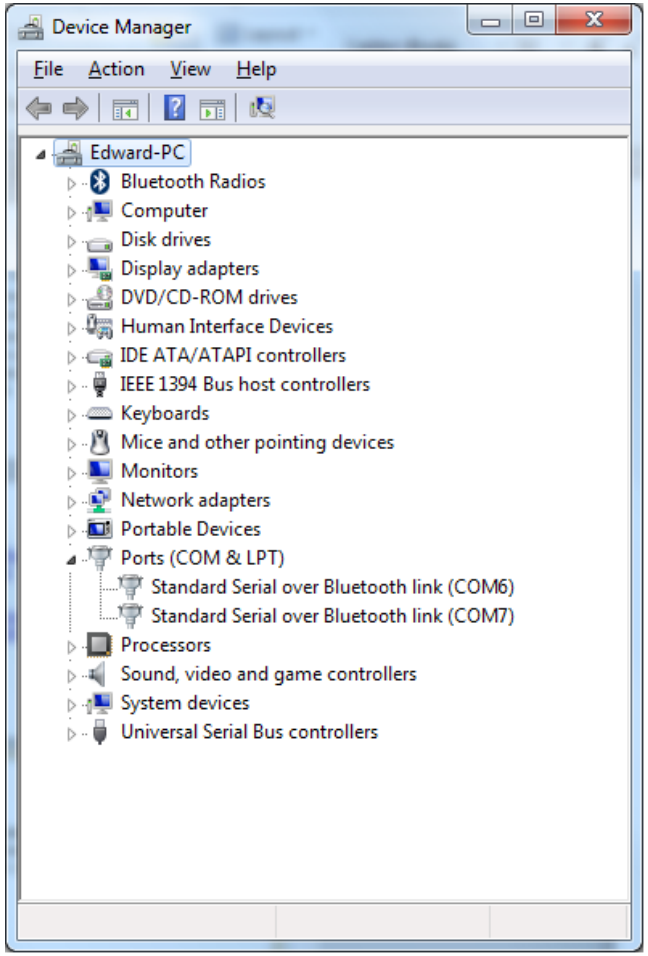
iWrap API

- http://bluegiga.com/as/current/doc/html/
- http://www.kelag.ch/industriesensoren/b

Serial Port Profile

- It emulates a serial cable to provide a simple substitute for existing RS-232, including the familiar control signals.

Windows Device Manager



Bluetooth Client Code

```
void config_bt(){
  Serial.println("SET BT PAGEMODE 3 2000 1");
  Serial.println("SET BT NAME BT_ArduinoFirefly");
  Serial.println("SET BT ROLE 0 f 7d00");
  Serial.println("SET CONTROL ECHO 0");
  // to pair with arduinobt need auth off, no passcode
  Serial.println("SET BT AUTH *");
  Serial.println("SET CONTROL ESCAPE 43 00 0");
  Serial.println("SET CONTROL BAUD 115200,8n1");
  delay(200);
  Serial.flush();
  delay(200);
}
```

Bluetooth Host Code

```
void btConnect()
{
  Serial.println("SET CONTROL ECHO 0");
  Serial.println("SET CONTROL ESCAPE 43 00 0");
  Serial.println("SET PROFILE SPP ON");
  Serial.println("SET BT AUTH *");
  Serial.println("RESET");
  Serial.println("PAIR 00:06:66:04:af:9b");
  delay(4000);
  Serial.println("CALL 00:06:66:04:af:9b 1 RFCOMM");
  delay(5000);
  Serial.flush();
  delay(200);
}
```

Processing Code (Common)

```
void loop()
{
  receptionValue = LOW;

  if (Serial.available() > 0)
  {
    // read in status of other bluetooth device alarm
    // get incoming byte:
    inChar = Serial.read();

    if (inChar == '@')
    {
      receptionValue = HIGH;
#ifdef DEBUG
      digitalWrite(LED, HIGH); // set led HIGH
#endif
    }
    else
    {
      receptionValue = LOW;
#ifdef DEBUG
      digitalWrite(LED, LOW); // set led LOW
#endif
    }
    delay(200);
  }
```

Processing Code (cont)

```
 // read the value from the sensor:
sensorValue = calculateAverageTemperature();

#ifdef DEBUG
  Serial.println(sensorValue, DEC);
  Serial.println(threshold, DEC);
#endif

  if (sensorValue > threshold)
  {
     // send alarm to other bluetooth device(s)
     // multiplexing support?
     Serial.print("&");
  }
  else
  {
     // indicate no alarm to other bluetooth device(s)
     // multiplexing support?
     //Serial.print("A");
  }
```

Processing Code (cont)

```
if ((sensorValue > threshold) || (receptionValue == HIGH))
 {
    // turn the ledPin on
    digitalWrite(ledPin, HIGH);

    // emit sound on piezo element sensor
    for (long i = 0; i < 2048 * 1; i++ )
    {
       // 1 / 2048Hz = 488uS, or 244uS high and 244uS low to create 50% duty cycle
       digitalWrite(piezoPin, HIGH);
       delayMicroseconds(122); //244);
       digitalWrite(piezoPin, LOW);
       delayMicroseconds(122); //244);
    }

    delay(500);
    // turn the ledPin off:
    digitalWrite(ledPin, LOW);
    delay(500);
 }
}
```

Bluetooth Debugging

```
void debug_loop()
{
  // if we get a valid byte, read analog ins:
  if (Serial.available() > 0) {
    inByte = getbyte();  // get incoming byte
    if (inByte == '&' ) { // look for a &
                            Serial.print("Got an &  ");
                          infoSize = getInfo();
                            Serial.println("Done");
    }
    else if (inByte == '@' ) { // look for a 0
                          digitalWrite(ledPin, LOW); // set led LOW
                          Serial.print("Get string:  ");
                          for(int i=0;i<infoSize;i++)
                          {
                            Serial.print(EEPROM.read(i));
                          }
                          Serial.println();
                          Serial.print("Cleared string  size: ");
                          Serial.println(infoSize);
    }
  }
}

char getbyte()
{
  while (Serial.available() == 0) { //look for aviable data
    // do nothing, wait for incoming data
  }
  return Serial.read(); //return data if aviable
}
```

Bluetooth Debugging (cont)

```
int getInfo()
{
  int j=0;
  digitalWrite(ledPin, HIGH); // set led HIGH
  delay(2000);
  Serial.print("+++"); // switch to command mode
  delay(2000);

  Serial.println("INQUIRY 5 NAME");
 Serial.println("SET BT AUTH *");
  delay(2000);
  Serial.println("PAIR 00:06:66:04:af:9b");
  delay(4000);
 Serial.println("CALL 00:06:66:04:af:9b 1 RFCOMM");
  delay(5000);
 Serial.println("LIST");

  for (int i=0; i <= 10; i++){
    delay(1000);
    while (Serial.available() > 0 && j <512) {
                        inByte = getbyte();  // get incoming byte
                        EEPROM.write(j, inByte);
                        j++;
    }
    delay(1000);
  }
  delay(2000);
  Serial.print("+++"); // switch to data mode
  delay(2000);
  digitalWrite(ledPin, LOW); // set led low
  return j;
}
```

Command and Data Modes

- SET CONTROL ESC to modify mode switching sequence.
- Default mode switch is “+++”
- Command mode to process iWRAP commands.
- Data mode to process data.

Future Enhancements

- Multiplexing mode “SET CONTROL MUX”
- The advantage of this multiplexing mode is that several *Bluetooth* connections can be handled simultaneously and there is no need to do time consuming data-command-data mode switching. However the downside is that the performance of iWRAP is reduced, since the firmware needs to handle the multiplexing protocol and the overhead it causes.
- Hex sequences rather than ascii commands.

Future Enhancements

- Personal Area Networking (PAN)
- IP over bluetooth, support not stated in iWRAP documentation. Possible?

Other Options and Issues

- Convert Arduino C to AVR C.
- Piezo element higher volume.
- ArduinoBT subject to periodic temperature spike. Due to method used to apply power or Bluetooth processing. Bluetooth Firefly does not experience this issue.