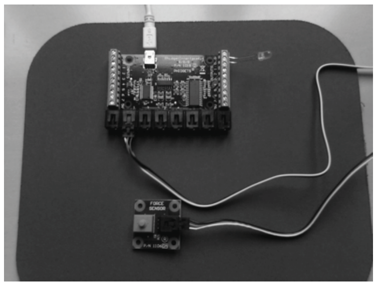
1.00 Lecture 25

Introduction to Sensors (Phidgets)

Reading for next time: Phidgets documentation

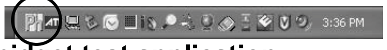
Phidget Interface Kit

1. Download the Phidgets software for your OS from www.phidgets.com/drivers.php.
2. Install it. Choose 32 bit or 64 bit version to match your OS and the version of Java you installed
3. Download the phidget .jar file from www.phidgets.com/programming_resources.php under Java
4. Unzip it to someplace where you can find it again.
5. Open your Phidget kit and find:
   1. USB cable to connect the interface board to your computer
   2. Interface board (1018)
   3. Force sensor (1106) with its cable
   4. A green LED
6. Connect them as in the image
   - USB from laptop to interface board
   - Force sensor to Analog In1
   - LED wired between GND and Digital Out 1
   - Short wire in GND, long in Digital Out 1
   - Use the screwdriver for the LED

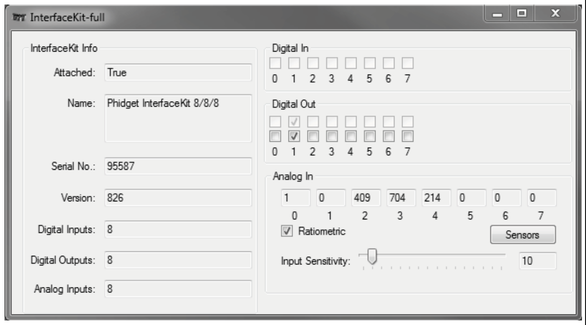


Phidget Interface Kit, 2

- If you have installed the Phidget software, you should see a Phidget icon in your taskbar:



- Click it. It should bring up the Phidget test application
  - If it brings up the Phidget control panel, click the General tab and then double click the Phidget interface kit device to bring up the test app
- Press the Phidget force sensor button. Watch the reading change.
- Click Digital Out box 1
- LED should light up



InterfaceKit-full

InterfaceKit Info

Attached:	True
Name:	Phidget InterfaceKit 8/8/8
Serial No.:	95587
Version:	826
Digital Inputs:	8
Digital Outputs:	8
Analog Inputs:	8

Digital In

☐	☐	☐	☐	☐	☐	☐	☐
0	1	2	3	4	5	6	7

Digital Out

☐	✓	☐	☐	☐	☐	☐	☐
☐	✓	☐	☐	☐	☐	☐	☐
0	1	2	3	4	5	6	7

Analog In

1	0	409	704	214	0	0	0
0	1	2	3	4	5	6	7

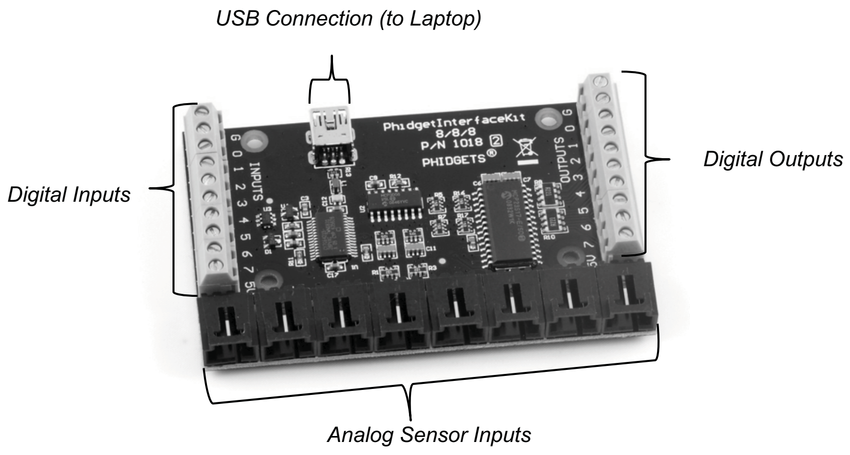
✓ Ratiometric

Sensors

Input Sensitivity: 10

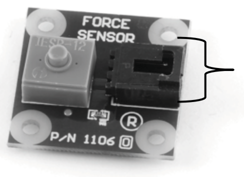
Courtesy of Phidgets. Used with permission.

Phidget Interface Anatomy



Courtesy of V. Judson Harward. Used with permission.

Phidget Force Sensor (Button)

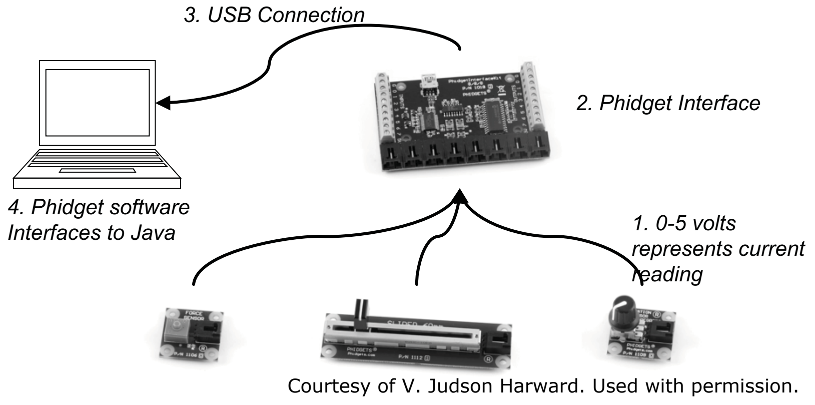


3 wire connector to:
interface card:
GND
+5V
Sensor out, 0-5V

Courtesy of V. Judson Harward. Used with permission.

- **Sensor outputs a voltage from 0 to 5 volts that is proportional to the force on the button**
- **Converted to a digital value by the interface card and relayed to the laptop over USB**
- **Most Phidget sensors work this way**

Phidget Architecture


3. USB Connection
2. Phidget Interface
4. Phidget software Interfaces to Java
1. 0-5 volts represents current reading

Courtesy of V. Judson Harward. Used with permission.

0. Multiple Sensors

Phidgets

Sensor	O		Units
1112 – slider (60mm)	0 – 1000	0= left 1000= right	mm= 0.06*s
1109 – rotation	0 – 1000	0= 0° 1000= 300°	degrees= 0.3*s
1127 – light	1 – 950 approx	1= moonlight, 1000= TV studio	lux= s
1110 – touch	0 – 1000	0= touch, <u>or</u> 1000= no touch	Yes/no only
1106 – force	0 – 1000	0= no force, <u>to</u> 1000= max	Not accurate enough to measure force
1124 - temperature	-30°C - +80°C	°C	temperature= 0.2222*s -61.111
1108 – magnetic	0 – 1000	gauss	ϕ(G)= 500 – s
1102 – IR reflective (at 3 to 7mm)	0 – 1000	s<400: no object s>= 400: object	Yes/no only

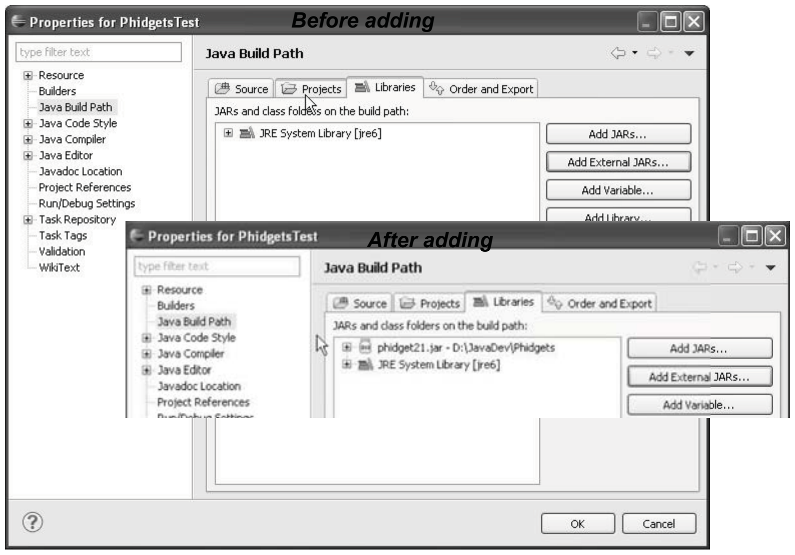
Phidgets technology

Sensor	Technology
1112 – slider (60mm)	Linear potentiometer
1109 – rotation	Potentiometer
1127 – light	*NPN transistor*
1110 – touch	Capacitive change sensor; will work thru 1/8” glass, plastic or paper
1106 – force	*Piezoelectric*
1124 – temperature	*Silicon diode*
1108 – magnetic	Linear Hall effect
1102 – IR reflective (at 3 to 7mm)	Infrared emitting diode, phototransistor

Phidgets and Java

- Download PressureController.java and compile it in a new project
  - You will get errors because Eclipse can't find the Phidget .jar file, the library that tells Java how to communicate with Phidgets
- Open the Java Properties/Java Build Path popup by right clicking on the project
- Click "Add External Jars…" and navigate to where you unzipped the phidget21.jar file
- Select it and click Open, and then OK
  - Next slide shows before and after shots
  - Errors will disappear from java files
- Run PressureController

Phidgets and Java

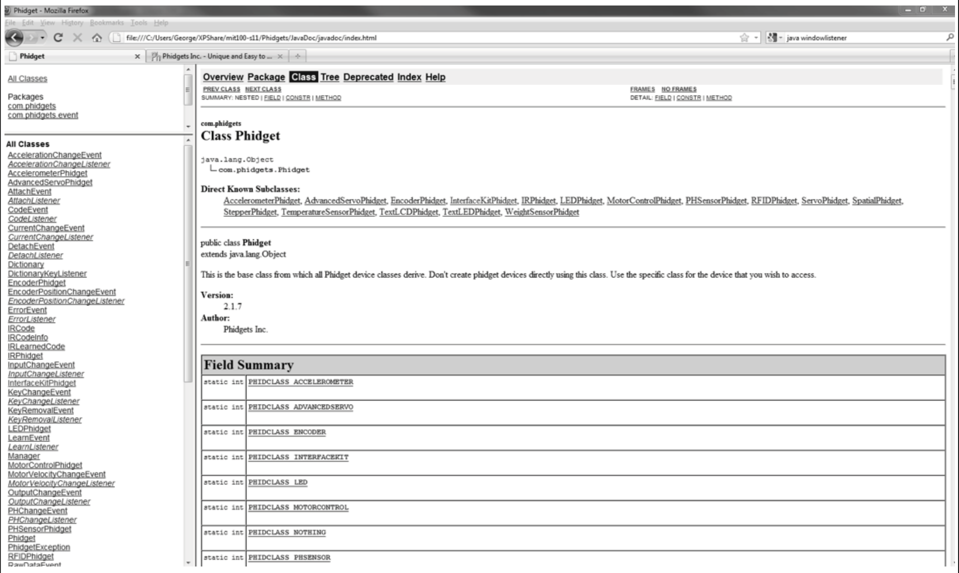


Courtesy of The Eclipse Foundation. Used with permission.

Phidget21.jar

- Jar file is a Java archive
  - Zip format of compiled (byte code) Java classes
- By placing it in your project, you can use all its classes
- See its documentation for a list of classes and methods. Download from
  - phidgets.com/programming_resources.php
  - Unzip
  - Bookmark it in your browser

Phidgets Javadoc



Phidget - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Users/George/XPShare/mit100-s11/Phidgets/JavaDoc/javadoc/index.html
java windowlistener
Phidget
Phidgets Inc. - Unique and Easy to ...
Overview Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

All Classes

Packages
com.phidgets
com.phidgets.event

All Classes
AccelerationChangeEvent
AccelerationChangeListener
AccelerometerPhidget
AdvancedServoPhidget
AttachEvent
AttachListener
CodeEvent
CodeListener
CurrentChangeEvent
CurrentChangeListener
DetachEvent
DetachListener
Dictionary
DictionaryKeyListener
EncoderPhidget
EncoderPositionChangeEvent
EncoderPositionChangeListener
ErrorEvent
ErrorListener
IRCode
IRCodeInfo
IRLearnedCode
IRPhidget
InputChangeEvent
InputChangeListener
InterfaceKitPhidget
KeyChangeEvent
KeyChangeListener
KeyRemovalEvent
KeyRemovalListener
LEDPhidget
LearnEvent
LearnListener
Manager
MotorControlPhidget
MotorVelocityChangeEvent
MotorVelocityChangeListener
OutputChangeEvent
OutputChangeListener
PHChangeEvent
PHChangeListener
PHSensorPhidget
Phidget
PhidgetException
RFIDPhidget
RawDataEvent

com.phidgets
Class Phidget

```
java.lang.Object
  └ com.phidgets.Phidget
```

Direct Known Subclasses:
AccelerometerPhidget, AdvancedServoPhidget, EncoderPhidget, InterfaceKitPhidget, IRPhidget, LEDPhidget, MotorControlPhidget, PHSensorPhidget, RFIDPhidget, ServoPhidget, SpatialPhidget, StepperPhidget, TemperatureSensorPhidget, TextLCDPhidget, TextLEDPhidget, WeightSensorPhidget

public class **Phidget**
extends java.lang.Object

This is the base class from which all Phidget device classes derive. Don't create phidget devices directly using this class. Use the specific class for the device that you wish to access.

Version:
2.1.7
Author:
Phidgets Inc.

Field Summary	
static int	PHIDCLASS_ACCELEROMETER
static int	PHIDCLASS_ADVANCEDSERVO
static int	PHIDCLASS_ENCODER
static int	PHIDCLASS_INTERFACEKIT
static int	PHIDCLASS_LED
static int	PHIDCLASS_MOTORCONTROL
static int	PHIDCLASS_NOTHING
static int	PHIDCLASS_PHSENSOR

Courtesy of Phidgets. Used with permission.

Phidgets programming

- Phidgets are designed to be used in an event-driven architecture
  - They report their values when they change
  - They should not be polled (inefficient)
- Phidgets can be used in a Web architecture
  - Phidgets on one computer can be accessed from other computers via Web services
    - Web services are essentially remote method calls
    - We don't use this feature in 1.00
- Every Phidget has a unique serial number
  - Can use this to identify each device

Exercise 1

- Exercise 1 is model-view-controller
  - No model yet, since we're just displaying data
- PressureController is JFrame, as usual. It doesn't do anything yet; you'll complete it.
  - One data member so far: Phidgets interface object
  - Constructor: adds window listener to close Phidgets interface object when JFrame closed
  - main() creates GUI object, calls openIntfcKit()
  - openIntfcKit():
    - Adds error listener and sensor change listener
    - "Opens" sensor, waits for it to "attach"
    - Sets sensor "ratiometric": has proportional output
  - closeIntfcKit(): closes Phidgets interface object
- PressureView: JPanel. No changes needed.
  - Draws rectangle proportional to pressure

PressureController.java, 1

```
import com.phidgets.*;
import com.phidgets.event.*;
import java.awt.event.*;
import javax.swing.*;

public class PressureController extends JFrame {
   private InterfaceKitPhidget interfaceKit;  // Core Phidget sw

   public static void main(String[] args) {
     PressureController pc = new PressureController();
     pc.setSize(100, 100);    // Blank GUI for now
     pc.setVisible(true);
     pc.openIntfcKit();       // Code on next page
   }
   // Constructor just listens for JFrame closing event
   public PressureController() { // Close Phidget intfc kit on exit
     // WindowListener interface has 7 methods
     // WindowAdapter has empty method bodies for all
     // Override only the needed one(s)
     addWindowListener(new WindowAdapter() {
       public void windowClosing(WindowEvent we) {closeIntfcKit();}
     });
   }
```

PressureController.java, 2

```
private void openIntfcKit() {
  try {
    interfaceKit = new InterfaceKitPhidget();
    interfaceKit.addErrorListener(new ErrorListener() {
      public void error(ErrorEvent ee) {
        System.out.println("Error event for " + ee); }
    });

    interfaceKit.addSensorChangeListener(new SensorChangeListener() {
      public void sensorChanged(SensorChangeEvent se) {
        System.out.println(se); }
    });

    interfaceKit.openAny();            // Open first sensor found
    System.out.println("Waiting for PressureSensor attachment...");
    interfaceKit.waitForAttachment();  // Wait for it to be available
    interfaceKit.setRatiometric(true);
    while (!interfaceKit.getRatiometric());  // Confirm ratiometric
     } catch (PhidgetException pe) {
           System.err.println(pe);
     }
 }
```

PressureController.java, 3

```
   private void closeIntfcKit() {
        System.out.println("Closing...");
        try {
            interfaceKit.close();
        } catch (PhidgetException pe) {
            System.err.println(pe); }
        interfaceKit = null;
        System.exit(0);
   }
}
```

PressureView

```
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;

public class PressureView extends JPanel {
   private PressureController controller; //Reference to controller
   public PressureView( PressureController c ) {
     controller= c;
     setBackground(Color.BLUE);
     setPreferredSize(new Dimension(300,300));
   }

   public void paintComponent( Graphics g ) {
     super.paintComponent( g );
     Graphics2D g2= (Graphics2D) g;
     double x= 100;
     double height= ((double) controller.getPressure()/1000.0) * 300;
     double width= 10;
     double y= 300 - height;
     Rectangle2D.Double rect= new Rectangle2D.Double(x,y,width,height);
     g2.setPaint( Color.red );
     g2.fill( rect );
}  }
```

Exercise 1, part 1

- Modify PressureController:
  - Add 3 private data members:
    - pressure (int)
    - index (location) of pressure sensor (int), equals 1
    - PressureView object pv
  - Main(): replace setSize() with pack()
  - Constructor: Add 3 lines:
    - Create PressureView object. It has the PressureController object as its argument. (Use `this`)
    - Call getContentPane, add PressureView object to center of pane
  - Write getPressure() method, which just returns pressure
- Compile this.
  - It won't run yet; there is one more step

Exercise 1, part 2

- Complete PressureController:
  - Modify sensorChanged() in openIntfcKit():
    - In the provided code, sensorChanged() just prints out the pressure. Remove this line.
    - Note that se is the SensorEvent object
    - Remember you connected the force sensor to input 1
    - If se.getIndex()== 1 (event comes from input 1) then…
      - Call se.getValue(), which returns int (0-1000), to set pressure
      - Call pv.repaint(), where pv is the PressureView object
      - (PressureView will ask controller for pressure value as part of repainting)
  - Compile and run.
    - Always close all previous runs, and always close the Phidgets test application. If there is more than one application looking for Phidgets events, they get confused.

Exercise 2, part 1: LED

- Modify `PressureController` to use the LED at position 1 to indicate that the pressure applied has crossed a threshold (=50)
  - New data members: LEDIndex= 1 and threshold= 50
  - You may initialize these or set via constructor arguments
- Compile but don't run it yet.

Exercise 2, part 2: LED

- In `sensorChanged()`:
  - Write an if-else statement to:
    - Turn the LED on if pressure is over threshold, or
    - Turn the LED off if pressure is under threshold
  - Use `setOutputState( int outIndex, boolean onOrOff)` method `from InterfaceKitPhidget`
  - You must place the if-else in a try-catch block to catch `PhidgetException pe`:

```
catch(PhidgetException pe)
      {System.err.println(pe);}
```

- Compile but don't run it yet.

Exercise 2, part 3: LED

- In `openIntfcKit()`:
  - Use `setOutputState()` to make sure LED is off at start of program. Do it near the end of the method.
- In `closeIntfcKit()`:
  - Use `setOutputState()` to make sure LED is off at end of program.
- Compile and run. Close all previous runs.
  - This should now work, and pressing the button should turn on the LED.