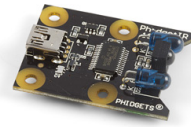


1055 User Guide



Go to this device's product page ^[1]

Getting Started

Checking the Contents

You should have received: **In order to test your new Phidget you will also need:**

- A PhidgetIR Board
- A Mini-USB Cable
- A mounting hardware kit
- Any IR Remote control
- Something that's controlled via IR


Connecting the Pieces

Connect the PhidgetIR to your computer using the Mini-USB cable.




Testing Using Windows 2000 / XP / Vista / 7

Make sure you have the current version of the Phidget library installed on your PC. If you don't, follow these steps:

1. Go to the Quick Downloads section on the Windows page
2. Download and run the Phidget21 Installer (32-bit, or 64-bit, depending on your system)
3. You should see the  icon on the right hand corner of the Task Bar.


Running Phidgets Sample Program

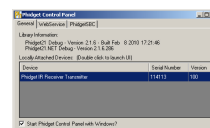
Double clicking on the  icon loads the Phidget Control Panel; we will use this program to ensure that your new Phidget works properly.

The source code for the **IR-full** sample program can be found in the quick downloads section on the C# Language Page. If you'd like to see examples in other languages, you can visit our Languages page.

Updating Device Firmware

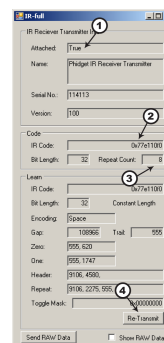
If an entry in this list is red, it means the firmware for that device is out of date. Double click on the entry to be given the option of updating the firmware. If you choose not to update the firmware, you can still run the example for that device after refusing.

Double Click on the  icon to activate the Phidget Control Panel and make sure that the **Phidget IR Receiver Transmitter** is properly attached to your PC.

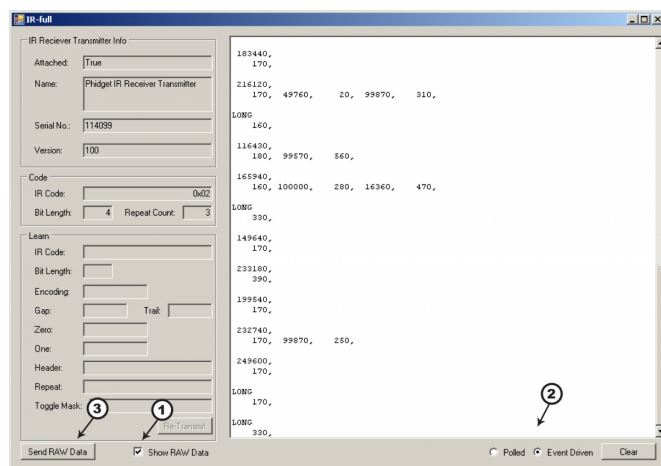


Learn Mode

1. Double Click on Phidget IR Receiver Transmitter in the Phidget Control Panel to bring up IR-full and check that the box labelled Attached contains the word True, as in the image to the right.
2. Point your remote at the PhidgetIR and press a button. If a code is received, it will be shown in the code box.
3. Press and hold a button. The repeat count should start to count up. After about 4 repeats, the Learn section should be filled in - the code has now been learned by the PhidgetIR. When in learning mode, try to minimize possible interference from other sources such as fluorescent lights.
4. Press the ReTransmit button to retransmit the code. It will function just as if you were pushing the same button on the actual remote.



Raw Data Mode



1. If the PhidgetIR cannot decode the data automatically, you can access the raw data stream directly.
2. You can access this data using an event or a polling function.
3. For the purpose of the example, the raw data that is sent by pushing the SendRAWData button will adjust the volume on a Mac Laptop computer.

Testing Using Mac OS X

1. Go to the Quick Downloads section on the Mac OS X page
2. Download and run the Phidget OS X Installer
3. Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
4. Make sure that the Phidget IR Receiver Transmitter is properly attached.
5. Double Click on Phidget IR Receiver Transmitter in the Phidget Preference Pane to bring up the IR-full Sample program. This program will function in a similar way as the Windows version.

Using Linux

For a step-by-step guide on getting Phidgets running on Linux, check the [Linux page](#).

Using Windows Mobile / CE 5.0 / CE 6.0

For a step-by-step guide on getting Phidgets running on Windows CE, check the [Windows CE page](#).

Technical Details

Consumer IR

The PhidgetIR can send and receive data encoded in various fashions as pulses of infrared light. The various encoding that the PhidgetIR supports are grouped under the general term 'Consumer IR' or CIR. CIR is generally used to control consumer products such as TVs, DVD players, etc. with a wireless remote control, but in general can be used for any application that needs to transmit low speed data wirelessly.

CIR is a low speed protocol - commands generally contain no more than 32-bits of data with a bit rate of at the most 4000 bits/second, but usually much less. There is no concession for anti-collision, so only one code can be transmitting at any time. Transmission distance depends on the power of the transmitter, but needs to be line of sight - though generally this can include bouncing off walls/ceilings, etc.

CIR data is transmitted using a modulated bit stream. Data is encoded in the length of the pulses / spaces between pulses, of IR light. The pulses of IR light are themselves modulated at a much higher frequency (usually ~38kHz) in order for the receiver to distinguish CIR data from ambient room light.

Further Reading

You can read more about IR remote controls in our [IR Remote Control Primer](#).

API

We document API Calls specific to this product in this section. Functions common to all Phidgets and functions not applicable to this device are not covered here. This section is deliberately generic. For calling conventions under a specific language, refer to the associated API manual in the [Quick Downloads](#) section for that language. For exact values, refer to the device specifications.

Data Structures

```
IR_CodeInfo {  
    int dataLength;  
    IR_Encoding encoding;  
    IR_Length length;  
    int gap;  
    int trail;  
    int header[2];  
    int one[2];  
    int zero[2];  
    int repeat[MAX_REPEAT_LENGTH];  
    int min_repeat;  
    byte toggle_mask[MAX_DATA_LENGTH];  
}
```

```

    int carrierFrequency;
    int dutyCycle;
}

```

The CodeInfo Structure/Class represents all aspects of a code except for the data. This is used when learning and when transmitting codes.

- **dataLength:** The length of the code data in bits
- **encoding:** The encoding used for the data. Possible values are: Space, Pulse, Bi-Phase, RC5 and RC6
- **length:** Whether the code length (including gap) is constant or variable.
- **gap:** The gap time. For constant length codes this is the entire code time including the data and the gap, for variable length codes, this is the gap time only.
- **trail:** Trailing bit time.
- **header:** Header pulse and space. Specify 0 for no header.
- **zero:** The pulse and space used to identify a '0' in the bit stream.
- **one:** The pulse and space used to identify a '1' in the bit stream.
- **repeat:** The repeat code. This must start and end with a pulse and be terminated with a 0 (null).
- **min_repeat:** Minimum number of times to repeat the code.
- **toggle_mask:** The toggle mask. These bits are toggled when transmitting.
- **carrierFrequency:** The carrier frequency in Hz. Default is 38000. Range is 10kHz - 1MHz
- **dutyCycle:** The period of the carrier frequency. Default is 50. Range is 10-50.

Depending on the encoding used, most of these can be set to 0/null to select default values.

Functions

Transmit(Byte[] data, IR_CodeInfo codeInfo)

Transmits a code.

TransmitRepeat()

Transmits a repeat for the last transmitted code. Note that this function must be called before the gap time has elapsed since the original code transmission, or the repeat code will be transmitted too late.

TransmitRaw(int[] data, int offset, int count, int carrier, int dutyCycle, int gap)

Transmits a raw data stream of length count starting at offset, at a specific carrier frequency and duty cycle, with a microsecond gap after the data is sent. Carrier, duty cycle, gap, count, and offset can be set to 0 to specify default values. The data array must start and end with a pulse (have an uneven number of values).

int ReadRaw(int[] data, int length)

Reads up to a specified amount of raw data in the data buffer. If no data is available, returns 0, else returns number of ints read. Read data always starts with a space and ends with a pulse.

Byte[] LastCode() [get]

Gets the last code received. If there is no last code, this will throw the EPHIDGET_UNKNOWNVAL exception.

(Byte[], IR_CodeInfo) LastLearnedCode() [get]

Gets the last learned code. Returns both the code and a CodeInfo structure/object. If there is no last code, throws the EPHIDGET_UNKNOWNVAL exception.

Events

Code(Byte[] data, int bitLength, Bool repeat)

The code event is fired every time a code is received and correctly decoded. The data array has a length 1/8 of bitLength with the MSB being in position 0. MSB is considered to be the first bit received. Repeat will be true if a repeat is detected - either timing wise or via a repeat code. False repeats can happen if two separate button presses happen close together.

Learn(Byte[] data IR_CodeInfo codeInfo)

The Learn event fires when a button has been held down long enough for PhidgetIR to have learned the CodeInfo values - usually about 1 seconds, or after 4 repeats.

RawData(int[] data)

The Raw Data event will fire every time the PhidgetIR gets more data - at most once every 8ms.

Product History

Date	Board Revision	Device Version	Comment
May 2010	0	100	Product Release
May 2011	0	101	getLabelString fixed for labels longer than 7 characters

References

[1] http://www.phidgets.com/products.php?product_id=1055

Article Sources and Contributors

1055 User Guide *Source:* http://www.phidgets.com/wiki/index.php?title=1055_User_Guide *Contributors:* Burley, Mparadis

Image Sources, Licenses and Contributors

Image:1055.jpg *Source:* <http://www.phidgets.com/wiki/index.php?title=File:1055.jpg> *License:* unknown *Contributors:* Mparadis

File:1055_0_Connecting_The_Hardware.jpg *Source:* http://www.phidgets.com/wiki/index.php?title=File:1055_0_Connecting_The_Hardware.jpg *License:* unknown *Contributors:* Mparadis

File:Ph.jpg *Source:* <http://www.phidgets.com/wiki/index.php?title=File:Ph.jpg> *License:* unknown *Contributors:* Mparadis

File:1055_0_Control_Panel_Screen.jpg *Source:* http://www.phidgets.com/wiki/index.php?title=File:1055_0_Control_Panel_Screen.jpg *License:* unknown *Contributors:* Mparadis

File:1055_0_IR_Raw_Data_Screen.jpg *Source:* http://www.phidgets.com/wiki/index.php?title=File:1055_0_IR_Raw_Data_Screen.jpg *License:* unknown *Contributors:* Mparadis

File:1055_0_IR_Screen.jpg *Source:* http://www.phidgets.com/wiki/index.php?title=File:1055_0_IR_Screen.jpg *License:* unknown *Contributors:* Mparadis