

Interactive Art with the MIT Handy Board

Colby Leider

Department of Music, Dartmouth College

<http://music.dartmouth.edu/~colby/hb.html>

cnl@dartmouth.edu

Introduction

Artists' demands for innovative, application-specific control interfaces for computer music compositions and installations necessitate a powerful yet easy-to-use multipurpose system. The Handy Board, designed by Dr. Fred Martin of the MIT Media Laboratory as a 68HC11 development platform for educational robotics use at the university level, is well-suited for low- to medium-scale and budget-constrained projects, and is especially well-suited for a high school or undergraduate introductory course to electronic and computer music. It is easily interfaced with sensors and actuators and can transmit Musical Instrument Digital Interface (MIDI) data. It is inexpensive (available from various vendors for under \$300), battery-powered, lightweight, programmable in C and 68HC11 assembly, and widely used and supported. The Realtime MIDI Library for the Handy Board and accompanying documentation facilitate the rapid creation of unique and expressive custom controllers and installations with little or no prior software or hardware skills, making it ideal for educational use.

Figure 1 on the following page provides a system-level overview of the Handy Board in an interactive environment.

Typically, the user writes code in the Interactive C environment (discussed later) to process incoming data from sensors and downloads it from a computer to the Handy Board. (Other 68HC11 development tools may be used as well). Then, the Handy Board is

connected to sensors, actuators, and MIDI equipment and may be run from its own battery power without being connected to the computer.

Detailed information about the Handy Board is available online at the Handy Board home page (<http://lcs.www.media.mit.edu/groups/el/projects/handy-board/>), and information about the Handy Board Realtime MIDI Library and related topics is available online at <http://music.dartmouth.edu/~colby/hb.html>.

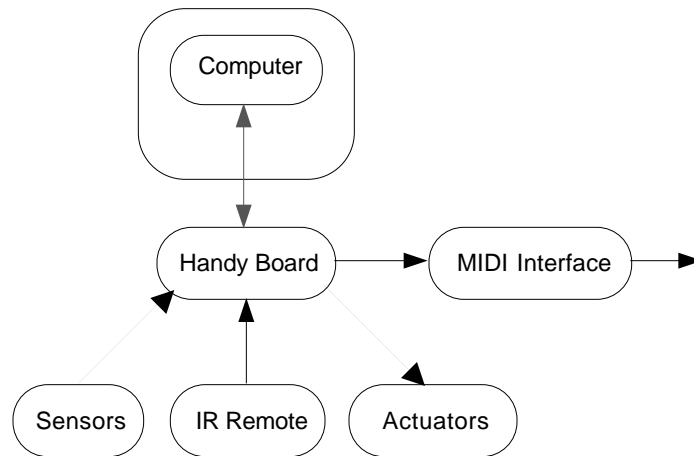


Figure 1: The Handy Board in an interactive environment.

Getting Started

The Handy Board may be either built from scratch using freely-distributable online schematics, purchased from one of several companies as a kit, or purchased preassembled and tested. Preassembled Handy Boards generally include their own startup guide, battery charger, and serial cable for programming via PC, Macintosh, or UNIX computers.

The creation of simple alternate controllers consists of three basic steps: (1) design of the hardware and software; (2) implementation of both; and (3) testing. The process of designing and implementing basic hardware for use with the Handy Board is outlined in the next section.

First, the user writes code in Interactive C on the host computer and downloads it

through a serial cable to the Handy Board. Then, the serial cable is disconnected from the host computer and plugged into a basic MIDI interface (for example, the JLC Cooper MacNexus or Opcode Studio 4), after which the Handy Board can send MIDI signals. The same serial port on the Handy Board is thus used for both receiving downloaded code from the host computer and transmitting MIDI information. Once the Handy Board is programmed, it no longer needs to communicate serially with the host, and it may operate in a configuration as illustrated in Figure 2.

Interactive C and the Realtime MIDI Library afford a great deal of control and formatting to incoming sensor data, and the intelligence behind a controller or installation may reside on the Handy Board's program memory (32K RAM). However, it is also possible to simply program the Handy Board to be a "dumb" data acquisition device and allow a program on a host computer (for example, a MAX patch) to control the intelligence of the installation or performance art piece. If MAX is used, the Serial object may be used instead of communicating via MIDI. In this case, the Handy Board may be employed in a system like that shown in Figure 3.

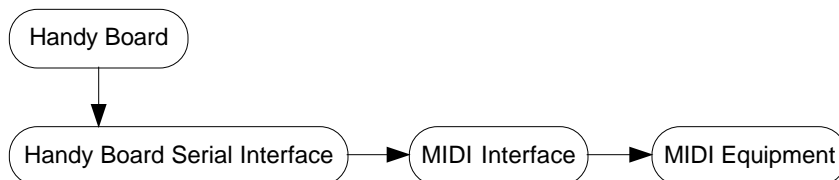


Figure 2: The Handy Board operating in a MIDI configuration.

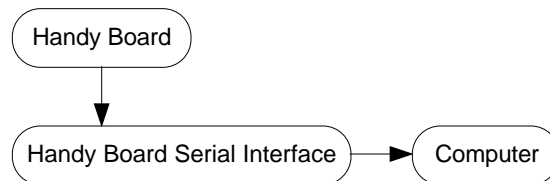


Figure 3: The Handy Board communicating serially with a computer.

Sensors

Connection of sensors to the Handy Board is trivial. Internal pullup resistors allow direct connect of photoresistors, temperature sensors, force-sensing resistors, potentiometers, etc., without extra circuitry. Linz [2] provides an excellent overview of various sensors useful in an interactive environment. Some basic types are listed below in Table 1.

Device	Example of Use
•Photoresistor	Sensing ambient light levels in a room to control realtime granular synthesis parameters in a realtime synthesis program, (e.g. SuperCollider or RTcmix)
•Temperature Sensor	Sensing temperature in an outdoor installation to control the overall volume of an algorithmically generated piece
•“Flex” Sensor	Attached to a regular hand glove to control MIDI pitch bend
•Sonar Ranging Device	Sensing a dancer’s movements on a stage (see Chabot [1])
•Infrared Remote Control	Buttons on the remote control change various MIDI parameters
•Force Sensing Resistor	Attached to an existing acoustical instrument to control realtime effects processing
•Potentiometer	Creating a knob in an installation that gives participants control of some parameter
•Infrared Detector	Close-range distance measurement
•Digital Compass	Adding position-proprioception to a mobile installation

Table 1: Basic Sensors for Use with the Handy Board

The Handy Board's built-in infrared receiver/transmitter allows a standard Sony-compatible remote control to interact with the Handy Board. It is possible to map each button of a remote control to control specific MIDI parameters using the Realtime MIDI Library (discussed in the next section). For example, the power button on a remote might issue an all notes off message, or pressing the volume buttons on the remote control might control MIDI main volume (controller number 7).

The Handy Board can also easily be connected to sonar rangefinding devices, such as the popular commercially-available Polaroid 6500. Interactive C code from Kent Farnsworth is available online at <http://lcs.www.media.mit.edu/groups/el/projects/handy-board/software/contrib/kent/hbsonar/> which interfaces with the Polaroid 6500 and provides basic functionality similar to that described by Chabot [1]. Furthermore, the Vector 2X digital compass may also be connected to the Handy Board for allowing a moving element in an installation to sense its own direction and generate responsive MIDI data. Information by Tom Brusehaver on interfacing this digital compass is available online at the following URL: <http://lcs.www.media.mit.edu/groups/el/projects/handy-board/software/contrib/tomb/>.

Realtime MIDI Library

The Handy Board Realtime MIDI Library consists of a collection of abstractions and functions written in Interactive C which help calibrate, scale, and format sensor inputs and produce MIDI messages. Interactive C is a portable subset of ANSI C that includes an integrated compiler/debugger and supports structures, multidimensional arrays, pointers, command-line execution (useful for debugging purposes, and multitasking (Martin [3]). Interactive C compilers exist for Macintosh, UNIX, and PC-compatible computers. The library provides an easy-to-use application framework for interactive art similar to that described by Linz [2].

For those wishing to use the Handy Board in standalone mode (with the intelligence of the interaction residing on the Handy Board), other functions (such as arpeggiators, loopers, harmonizers, etc.) may be easily written, and examples are provided as part of the

Realtime MIDI Library in `hb-midilib-misc`.

The Handy Board Realtime MIDI Library consists of eight files which are grouped according to the types of tasks the included functions perform. One of the files, `hb-midilib.lis`, is an Interactive C list file which contains the names of the other seven files which comprise the MIDI Library. Downloading `hb-midilib.lis` to the Handy Board automatically loads all the constituent files as well.

Occasionally it will be more convenient and efficient to copy some of the higher-level library functions that a program is using from the Realtime MIDI Library into the `main()` function itself, particularly if the function is to be called often. If speed is of particular importance for a project, the library function `turbo_on()` may be called, which allows the Handy Board to poll sensor inputs more often by disabling infrared decoding and pulse width modulation for motors.

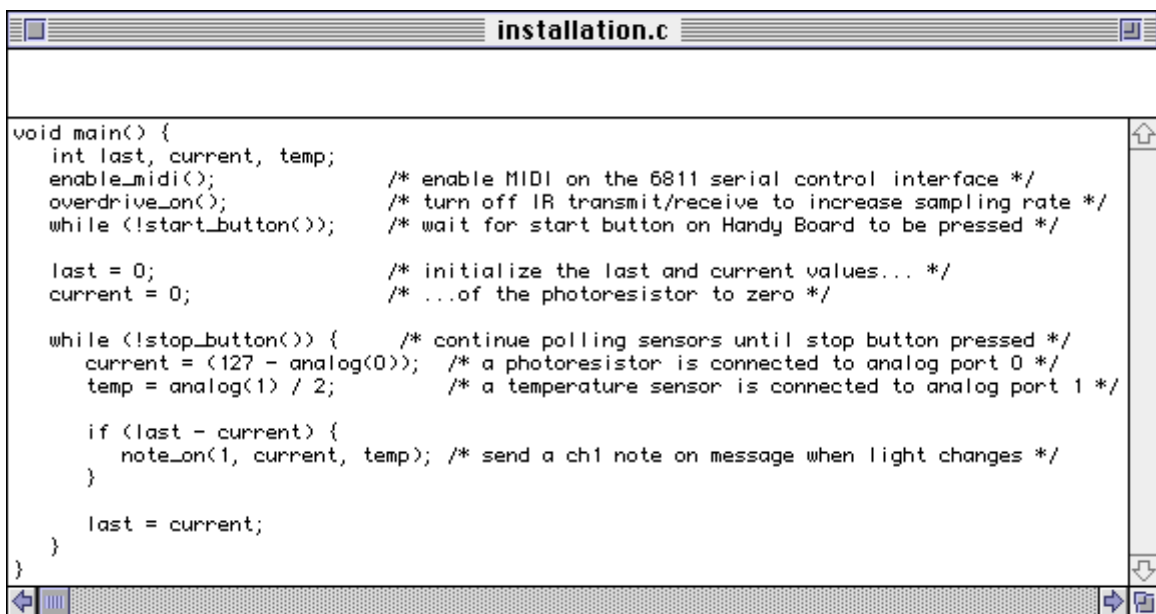
As noted, the Realtime MIDI Library is divided into seven files:

- (1) abstractions from the MIDI specification (`hb-midilib-defines.c`);
- (2) basic MIDI functions (`hb-midilib-midi.c`);
- (3) general serial I/O routines (`hb-midilib-serial.c`);
- (4) analog port routines (`hb-midilib-analog.c`);
- (5) LCD routines (`hb-midilib-lcd.c`);
- (6) basic arithmetic and array routines (`hb-midilib-math.c`); and
- (7) examples of higher-level MIDI functions (`hb-midilib-misc.c`).

The first file, `hb-midilib-defines.c`, contains abstractions from the MIDI specification in the form of C define statements (e.g., `#define MAIN_VOLUME 7`). The file `hb-midilib-midi` contains system-level functions which allow the Handy Board to transmit MIDI data, including implementations of the various MIDI channel voice messages (note on, note off, program change, etc.). The third file, `hb-midilib-serial.c`, contains general serial I/O routines after Randy Sargent of Newton Research Labs for allowing the Handy Board to send and receive serial data through the 68HC11 serial control interface

(SCI) port. These are useful for communicating directly to the MAX Serial object, for example. The fourth file, `hb-midilib-analog.c`, aids the user in scaling input voltages connected to the analog sensor ports to the desired range. The file `hb-midilib.lcd` contains LCD routines which aid the user in debugging programs by using the LCD to continually monitor sensor values. Basic arithmetic and array routines are contained in `hb-midilib-math.c` which allow the user perform simple operations such as finding array sums and arithmetic means. In the last file, `hb-midilib-misc.c`, several higher-level MIDI functions are provided as examples. A complete reference guide to all of the functions in the Realtime MIDI Library is available online at <http://music.dartmouth.edu/~colby/handbook.html>.

Programs continuously poll sensors and call functions of the library to generate the appropriate MIDI output stream. The Handy Board is also capable of interrupt-driven processing, although the current library uses polling to keep programming as simple as possible. Using the library, the user can map the input values from sensors attached to the Handy Board's analog and digital inputs and output the desired MIDI stream. Figure 4 illustrates code which drives an simple installation consisting of a sculpture equipped with a photoresistor and temperature sensor (about \$2 total).



```
void main() {
  int last, current, temp;
  enable_midi();          /* enable MIDI on the 6811 serial control interface */
  overdrive_on();         /* turn off IR transmit/receive to increase sampling rate */
  while (!start_button()); /* wait for start button on Handy Board to be pressed */

  last = 0;               /* initialize the last and current values... */
  current = 0;            /* ...of the photoresistor to zero */

  while (!stop_button()) { /* continue polling sensors until stop button pressed */
    current = (127 - analog(0)); /* a photoresistor is connected to analog port 0 */
    temp = analog(1) / 2;      /* a temperature sensor is connected to analog port 1 */

    if (last - current) {
      note_on(1, current, temp); /* send a ch1 note on message when light changes */
    }

    last = current;
  }
}
```

Figure 4: Code which drives a simple interactive installation.

The code in Figure 4 continuously polls the light and temperature sensors and sends a MIDI note on message to channel 1 whenever the ambient light level in the room changes. In this simple example, the light level controls the MIDI note number, and the temperature controls the MIDI note velocity.

When sensors are plugged into the Handy Board's digital port, voltages greater than about 2.5 volts register a logic high, while those below register a logic low. Thus the call `digital(1)` will return the value 1 when the sensor connected to digital port 1 reads a value greater than 2.5 volts, and it will return the value 0 when the sensor reads below 2.5 volts.

Sensors connected to the analog ports return values in the range 0–255 (with 0 corresponding to 0 volts and 255 corresponding to 5 volts). A call to the library function `display_analogs()` will continually display to the LCD screen the values associated with the analog ports. Using simple arithmetic, the values may be scaled and offset by a constant value to produce values in the desired range.

For example, in Figure 4 above, the variable `current`, which corresponds to the current light level, is set to `(127 - analog(0))` to ensure that it always contains a value in the desired range for controlling MIDI pitch. The variable `temp` is divided by two to return a useful range to control MIDI velocity. The scale and offset values are determined empirically through trial and error and depend on the characteristics of individual sensors. Different types of light and temperature sensors will require different scale and offset values.

A quick way to determine these values is to write a simple `main()` function which calls `display_analogs()`. By forcing a sensor to return its extreme values (for example, holding a book over a light sensor to block all light from its surface and then subsequently shining a bright light into it), and then noting the values printed to the LCD screen in each case, the optimal scale and offset values may easily be determined.

Alternatively, the library function `return_analog(0, 127, -1)` could have been called to set the value of `current`. In this case, 0 refers to analog port 0 (where the photoresistor is connected), 127 refers to the DC offset, and the sensor value is scaled by -1 .

A similar function from the Realtime MIDI Library may be called to test Sony-compatible infrared remote controls and integrate them into a project. Fred Martin and Brian

Silverman have written a driver which allows the Handy Board to decode these infrared signals, and it is available online at <http://lcs.www.media.mit.edu/groups/el/projects/handy-board/software>. To test the remote control, the user simply loads the file `sony-ir.icb` onto the Handy Board and enables it with the call `sony_init(1)`. Subsequent calls to `ir_data(1)` return the value received from the remote control. The Realtime MIDI Library function `ir_test()` automates the process of testing the integer values of each button on the remote control by continually polling for values and displaying them to the LCD screen. This way, the user may make a table of the received values for each button on the remote control (they do not always match—for example, “1” on the remote may transmit the value “0” to the Handy Board) and then assign specific MIDI functionality to each button.

One of the more useful features of Interactive C in terms of MIDI is its multitasking capabilities. Because of this, the process of polling each sensor input may have a unique process identification number. For example, a temperature sensor in an installation may only need to be polled every five minutes, whereas a light sensor may need to be polled 200 times per second.

Another benefit of the multitasking feature is that independent delay lines may be constructed. For example, sensor input into analog port 0 may be read for five seconds, then looped, while sensor input into analog port 1 may be read for ten seconds and looped. This is made possible by the fact that the polling of each of these sensors is given its own process, whereby delay (i.e., `sleep()` or `msleep()`) commands may be treated independently.

It is hoped that the Realtime MIDI Library will be a collaborative project, with users contributing their own functions for inclusion in the code repository at the project’s home page (<http://music.dartmouth.edu/~colby/hb.html>).

Hardware Expansion

The Handy Board can be expanded through the use of multiplexors which allow more sensors to be connected. A proposal for an expansion interface by Fred Martin which provides 24 additional analog inputs, along with more motor outputs and an onboard programmer for

Microchip Technology's PIC microcontrollers may be found on the World Wide Web at <http://lcs.www.media.mit.edu/groups/el/projects/handy-board/hbexp20b/index.html>.

Educational Uses

The Handy Board would work well in three types of classes in electronic and computer music. The first would be an introductory class in electronic and computer music without prerequisites. One Handy Board could serve the entire class, since sensors are easily attached and removed, thereby allowing students to schedule individual time to work. Students could undertake a guided two- or three-week interactive composition project using the Realtime MIDI Library and basic store-bought sensors—photoresistors, temperature sensors, Force-Sensing Resistors, potentiometers, etc. In such a project, students could add sensors to existing acoustic instruments or design simple new instruments and control MIDI synthesizers and effects units. For example, a small 0.3-inch Force Sensing Resistor attached to frog of a violin bow could control the reverberation time of an effects processor, or a series of pushbuttons placed across a floor could control MIDI program change information in a live performance composition.

Another class in which the Handy Board and Realtime MIDI Library would work well is a junior/senior-level project-oriented class in which students work on one of three types of projects: (1) an installation with motors, sonar or infrared proximity detectors, pressure pads, etc.; (2) an interactive dance piece; or (3) an interactive composition. In addition to controlling MIDI synthesizers and effects units, students with experience in computer music and programming could control audio synthesis parameters in real time in SuperCollider, RTcmix, or Csound.

The Handy Board could also be used with the MIDI Library in analog and digital circuits classes for musicians. Students could build their own hardware add-ons (such as sensor input multiplexors and multichannel infrared proximity detectors) for the Handy Board, as well build their own more complex musical instruments and installations.

The Handy Board accommodates various levels of technical skill: those with limited

exposure to programming and hardware can begin by writing simple programs that use the Realtime MIDI Library, and more technically advanced users can design and build additional hardware and write lower-level 68HC11 assembly drivers that communicate via MIDI.

Simple Design Examples

Two simple alternate controllers, Light-Sensitive Drum Sticks and the Pressure-Sensitive Data Glove, were created for performance in a concert at Dartmouth College.

Light-Sensitive Drum Sticks. Drum sticks are equipped with photoresistors on the tips. On one side of one of the sticks (where the player's thumb falls) is mounted a small Force-Sensing Resistor (FSR) from Interlink Electronics. Standard RJ11 telephone cable is soldered to the leads of the photoresistors and the FSR and connected to three of the analog inputs on the Handy Board. The user knob on the Handy Board is used as a calibration mechanism to control the light threshold level above which MIDI note on messages are sent and below which MIDI note off messages are sent. The drum sticks are played in the air in the vicinity of a light source (e.g., the monitor of a computer screen), and the thumb pressure on the FSR controls the MIDI volume level. Fewer than 40 lines of C code are required for basic MIDI functionality on this particular controller, and the total cost (apart from the cost of the Handy Board) is approximately \$11.

Pressure-Sensitive Data Glove. Force-sensing resistors are attached to the fingertips of a glove, and standard RJ45 cable is soldered to the FSR solder tabs and connected to the analog inputs of the Handy Board. Each finger is assigned a different MIDI channel, and MIDI data is transmitted every time the user touches a surface. For example, touching the index finger on a surface triggers a note on event on channel 1, after which the finger pressure is interpreted as modulation. Approximately 50 lines of C code are required for basic functionality, and the total cost (apart from the cost of the Handy Board) is approximately \$25.

Conclusions

The Handy Board and the Realtime MIDI Library are ideal for educational use and budget-constrained and portable interactive composition projects. Using the library, sensor data is easily formatted and mapped to MIDI data using simple, high-level C code with a minimum of technical knowledge or experience. While other such systems exist or can be created on a custom-made basis, Interactive C and the Realtime MIDI Library for the Handy Board afford a great deal of simplicity and user-friendliness in the creation of new interactive music projects.

Acknowledgments.

I would like to acknowledge the help and contributions of Kristine H. Burns, Victor Limary, Larry Polansky, Charles Dodge, Jon Appleton, Scott Lawrence, Leslie Stone, Kevin Parks, and Empi Esguerra and the support of the Bregman Electronic Music Studio.

Bibliography.

- [1] Chabot, X. "Gesture Interfaces and a Software Toolkit for Performance with Electronics", *Computer Music Journal*, Volume 14, Number 2, pp. 15-27.
- [2] Linz, R. 1996. "Towards the Design of a Real-Time Interactive Performance Sound System", *Leonardo Music Journal*, Volume 6, pp. 99-107.
- [3] Martin, F., et al. *Interactive C User's Guide*. http://www.newtonlabs.com/ic/ic_1.html.