Embedded Sound Synthesis

Victor Lazzarini, **Shane Byrne** and Joe Timoney

Maynooth University, Ireland

April 9, 2015

Introduction

This presentation will report on our experiences in building systems for embedded audio synthesis using the Intel Galileo computer.

We will start by describing the Galileo platform, and introduce its characteristics. This will be followed by a discussion of the software used.

We will finish by presenting a case study of a MIDI synthesizer using the Intel Galileo as the main component.

Galileo: GEN1

Galileo boards have been produced under two slightly different hardware configurations, namely, original (GEN1), and a revised specification (GEN2).

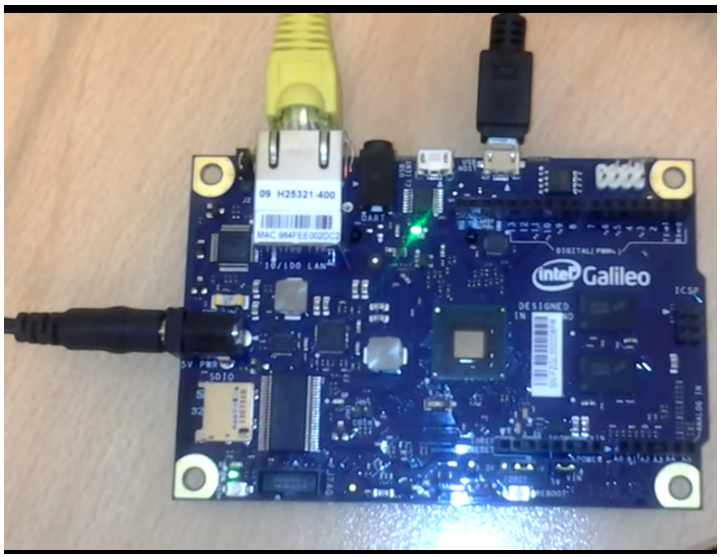


Figure 1: The Intel Galileo (GEN1) with ethernet and USB connections

Galileo: GEN2

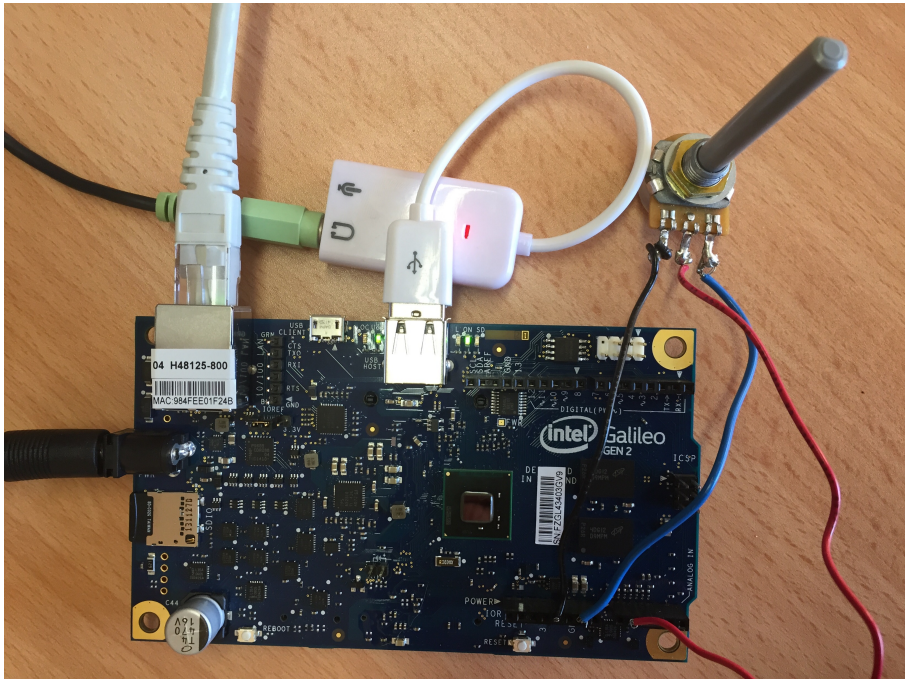


Figure 2: The Intel Galileo (GEN2) with ethernet, USB audio and a potentiometer connected to analog input 1 (pin A1)

Specifications

The two share some basic attributes that include:

- Quark processor (i586 instruction set), single core, 400 MHz ('Clanton')
- 10/100Mbit ethernet
- PCI Express
- USB 2.0 device and host interfaces
- microSD card reader.
- 3.5 mm RS-232 connector (GEN1) or 6-pin UART header (GEN2)

Arduino compatibility

20 General-Purpose Input/Output (GPIO) pins (6 multiplexed as analog inputs), plus power and Serial Peripheral Interface (SPI) headers.

- GEN1: external GPIO expander chip (Cypress CY8C9540A) is used to control most of the pins in the shield, with only two Quark GPIOs connected directly.
- GEN2: 12 Quark GPIOs fully accessible to the headers
- GEN1: Analog Devices AD7298 ADC IC (12-bit) analog input.
- GEN2: Texas InstrumentsADS108S102 IIO-ADC (10bit, scaled to 12bit)
- PWM (better resolution in GEN2)

System

There are two types of Linux images that are used in the Galileo, based on different versions of the standard C library. The system can be booted from flash or from the SD card:

- A smaller image, mostly meant to be run from the limited space in the board flash memory, built with uClibc.
- A larger image built with eglibc, more suitable for SD card-based installations with no size constraints.

Development environment

This is provided by the Yocto Project, which support the development of Linux distros for embedded systems, through its Board Support Package (BSP) for clanton. With this, we can build:

- a fully-functional customised Linux Standard Base distro for the board
- a Software Developer Kit (SDK) containing the gcc/g++ toolchain (and alsa)

A fully-functional port of Csound was built for the Galileo board using this cross-compilation environment.

Custom frontends

In order to access the basic Arduino-like functionality of the Galileo, specific frontends were developed: `gcsound` (GEN1) and `gcsound2` (GEN2).

Connections to the pins on the board are accomplished via the Linux Sysfs interface. This provides access to GPIOs via a number of files under `/sys/class/gpio` (for digital IO) and `/sys/bus/iio/devices/iio:device0/` (for analog IO).

Analog input

The analog inputs on the Galileo board are marked A0-A6. These inputs are offered to Csound orchestra in the software bus channels named as "analog*N*" where *N* is the analog port number.

```
ksig chnget "analog1"
```

The signal is delivered as a floating-point number normalised between 0 and 1 (corresponding to a 0 - 5V input range)

Digital input and output

The remaining 14 pins can be used for general-purpose digital input or output. Access is provided as requested, through blocking reading/writing operations. This functionality is implemented as new unit generators (opcodes) in the system:

```
ival   gpin inum
kval   gpin inum
       gpout ival, inum
       gpout kval, inum
```

where `ival` and `kval` are the GPIO values (0 or 1), and `inum` is the GPIO number.

Pin to GPIO mapping, Galileo GEN1

pin	mux selector, value	source/function
0	40, 0 40, 1	UART0 RXD (/dev/ttyS0) 50 (GPIO)
1	41, 0 41, 1	UART0 TXD (/dev/ttyS0) 51 (GPIO)
2	31, 0 31, 1	14 (Quark GPIO) 32 (GPIO)
3	30, 0 30, 1	15 (Quark GPIO) 18 (GPIO)
4	-	28 (GPIO)
5	-	17 (GPIO)
6	-	24 (GPIO)
7	-	27 (GPIO)
8	-	26 (GPIO)
9	-	19 (GPIO)
10	42, 0 42, 1	SPI1_CS (Quark) 16 (GPIO)
11	43, 0 43, 1	SPI1_MOSI (Quark) 25 (GPIO)
12	54, 0 54, 1	SPI_MISO (Quark) 38 (GPIO)
13	55, 0 55, 1	SPI_SCK (Quark) 39 (GPIO)

Pin access: GEN1

For the pins that require multiplexing, the gpout opcode will need to be used to select the correct source before accessing the pin from that source. For instance to access the GPIO for pin 0 and set it to 1, we have to use

```
gpout 1, 40
gpout 1, 50
```

so that GPIO 40 accesses the multiplex selector, selecting the source as GPIO 50, and we then set this to 1.

Pin to GPIO mapping, Galileo GEN2

pin	mux 1, value	mux 2, value	dir	22k res	source/function
0	-	-	-	-	UART0 RXD (/dev/ttyS0)
	-	-	32	33	11 (Quark GPIO)
1	45, 1	-	-	-	UART0 TXD (/dev/ttyS0)
	45, 0	-	28	29	12 (Quark GPIO)
2	77, 1	-	-	-	UART1 RXD (/dev/ttyS1)
	77, 0	-	34	35	13 (Quark GPIO)
	77, 0	-	-	35	61 (PCAL9535A GPIO)
3	76, 1	-	-	-	UART1 TXD (/dev/ttyS1)
	76, 0	64, 0	16	17	14 (Quark GPIO)
	76, 0	64, 0	-	17	62 (PCAL9535A GPIO)
4	-	-	36	37	6 (Quark GPIO)
5	66, 0	-	18	19	0 (Quark GPIO)
6	68, 0	-	20	21	1 (Quark GPIO)
7	-	-	-	39	38 (PCAL9535A GPIO)
8	-	-	-	41	40 (PCAL9535A GPIO)
9	70 0	-	22	23	4 (Quark GPIO)
10	70, 0	-	26	27	10 (Quark GPIO)
11	44, 1	72, 0	-	-	SPI_MOSI (spidev1.0)
	44, 0	72, 0	24	25	5 (Quark GPIO)
12	-	-	-	-	SPI_MISO (spidev1.0)
	-	-	42	43	15 (Quark GPIO)
13	46, 1	-	-	-	SPI_SCK (spidev1.0)
	46, 0	-	30	31	7 (Quark GPIO)

Pin access: GEN2

Example - to make the onboard led blink, we do:

```
instr blink
  kcnt init 0
  kLed init 0
  gpout 0, 46 ; select quark GPIO
  gpout 0, 30 ; set direction to output
  if kcnt == 100 then
     kLed = (kLed == 0 ?  1 : 0)
     gpout kLed, 7 ; turn led on and off
     kcnt = 0
   endif
  kcnt += 1
endin
```

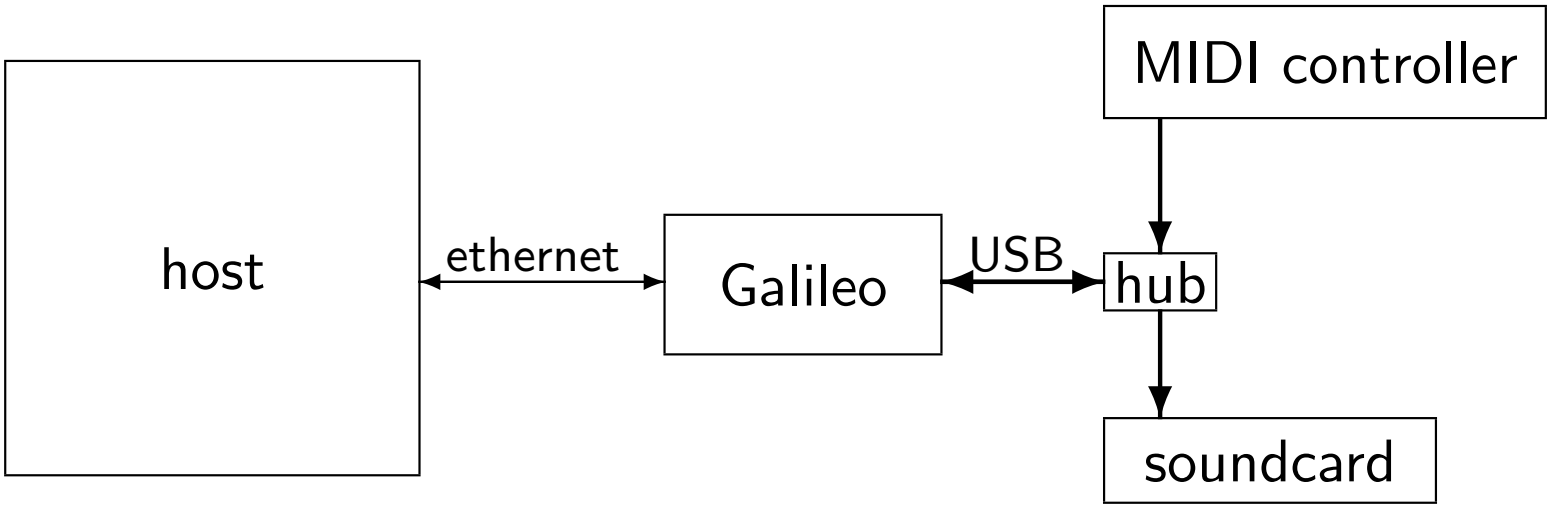
Description

As a case study to assess the potential of the Galileo board for music-making, a fully-fledged MIDI synthesizer was developed.

A Csound image for the board was developed, which includes a realtime preemptive kernel. This image can be simply copied into any microSD card compatible with the board (sizes between 2 and 32GB).

It contains the standard Csound command-line frontend `csound`, the custom frontends `gcsound` and `gcsound2`, the Csound library (plus some plugin opcodes), and the basic MIDI-based orchestra (midisynthesizer.csd). The image boots directly into the synthesizer process.

Flowchart



The host is not normally needed, as the synthesizer works without it. It is only required for debugging or for loading new instruments.

The instruments

The synthesizer contains three instruments, accessible via MIDI program change messages. The three instruments are:

1. Supersaw synth: a simple design based on five detuned sawtooth oscillators. Modulation controls detuning, CC 02 controls envelope attack, and CC 03 controls envelope release.
2. Pluck string: a Karplus-Strong-like instrument. Modulation controls brightness, CC 02 controls envelope attack, and CC 03 controls envelope release.
3. Voice: a ModFM formant synthesizer. Modulation controls vowel types, CC 02 controls attack, and CC 03 controls envelope release.

Demo video

Further possibilities

The technology described in this article has significant potential for further exploitation, beyond the simple case study discussed above. In particular, it has a direct application as a platform for Ubiquitous Music research and practice. Here are some examples:

- Portable live-electronics platform
- Programmable effects units
- Internet of Musical Things
- Low-cost cluster computing for audio

Conclusions

This paper reported on the implementation of an audio processing system using the Intel Galileo development board and Csound, running under a customised version of Linux for embedded devices.

While we have concentrated on a specific embedded platform, the ideas and principles discussed here can be applied elsewhere. In particular, we hope to develop similar systems for the Intel Edison in the near future.