Digital Synth Theremin using Arduino and PureData

Sandeep Dasari

May 11, 2018

Contents

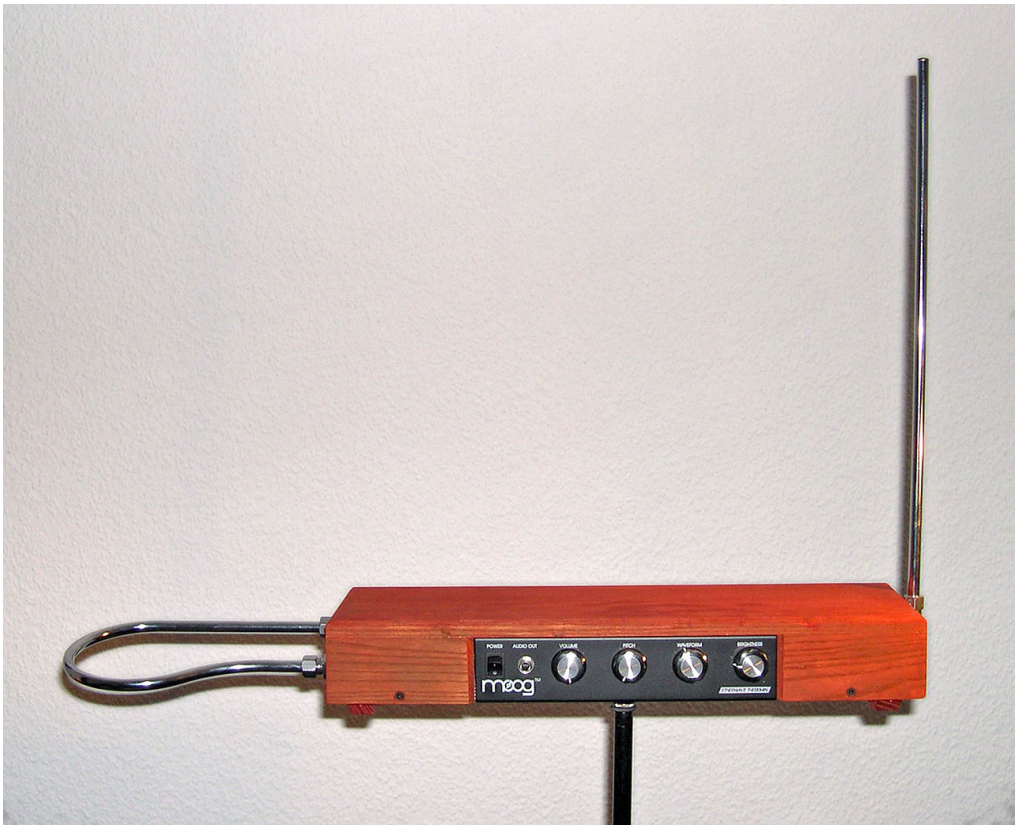

Figure 1: Etherwave MOOG Theremin

1 Abstract

Arduino UNO is a powerful open source development platform for developing real time machines that can be implemented in various fields. This project explores the field of music and digital signal processing to build a Digital Synth Theremin using PureData as a Digital Signal Processing platform. The Uno uses basic IR distance sensor for the operation of the instrument and the PureData to generate actual sound waves of different frequencies.

2 Proposed Idea

The initial motivation for this project came from digital synthesizer VSTs for laptops. These synthesizers are built to modulate a digital signal and change the way it sounds. This can be used to modulate any signal to generate interesting sounds of an instrument, in our case, a Theremin.

3 Introduction

The main aim of this project is to develop a Theremin that can run the signal into a DSP Program like Pure Data to modulate the sound frequency and handle all this in real time.

Theremin (Fig 1.) is an analog music instrument that can be played without contact. The instrument's controlling section usually consists of two metal antennas that sense the relative position of the thereminist's hands and control oscillators for frequency with one hand, and amplitude (volume) with the other. The electric signals from the

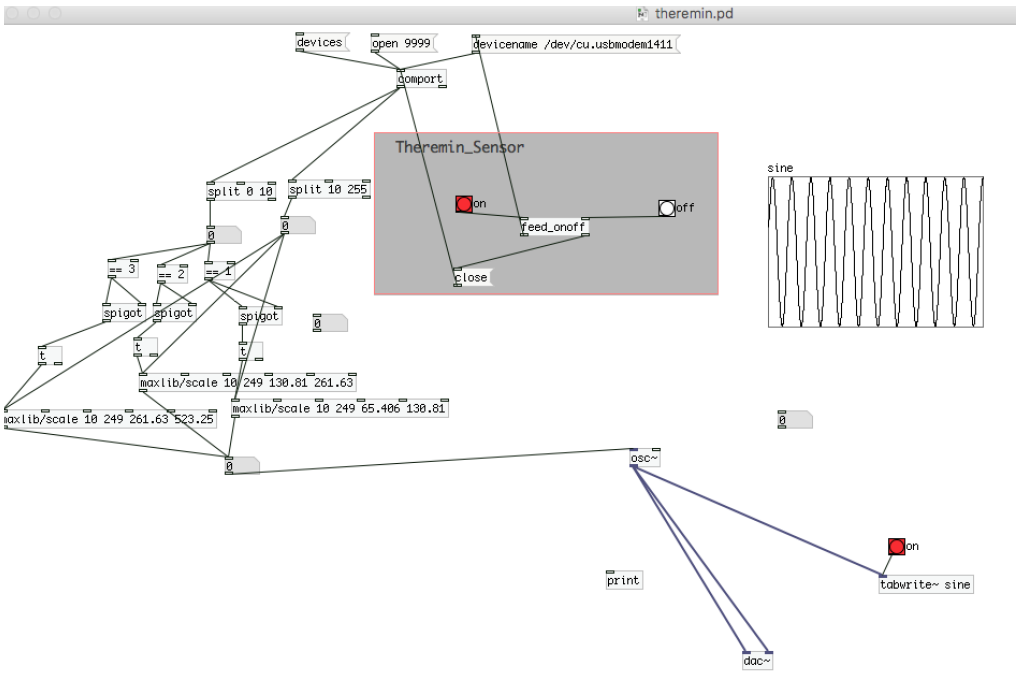

Figure 2: Pure Data theremin patch

theremin are amplified and sent to a loudspeaker.

This project tries to emulate an analog Theremin by connecting an Arduino to an IR distance sensor. The Sharp IR distance sensor used returns a voltage to Arduino, which is processed and converted into a map of frequencies. These frequencies are passed into Pure Data to generate a sine wave sound.

Pure Data (Pd) is a visual programming language developed by Miller Puckette in the 1990s for creating interactive computer music and multimedia works. While Puckette is the main author of the program, Pd is an open source project with a large developer base working on new extensions. It is released under a license similar to the BSD license. It runs on GNU/Linux, Mac OS X, iOS, Android and Windows.
Pure Data can be widely applied to generate, modulate and change an audio signal. In our case, we will be passing in an ADC converted voltage to pure data, that is mapped out into octaves of frequencies and passed into an oscillator. Pure Data has inbuilt objects for an oscillator (osc) that can be invoked to generate sound

4 Circuit and Hardware

The circuit of this instrument is based on **Arduino UNO**. The Arduino UNO has 6 analog pins. The IR Sensor will be connected to one of the analog pins. Since, Arduino UNO already has an onboard Analog to Digital Converter, there is no need to convert the analog signals to digital explicitly. The IR sensor has 2 other connections that go to the 5V power supply and the Ground.

The UNO will also have a button connected to one of the digital pins. This button will let the user toggle between the octaves of frequencies. The button is modelled to toggle between 3 octaves of frequencies starting from the lowest frequency of 130Hz which is the frequency of low C or C2 on a MIDI keyboard. The UNO will also a have USB cable plugged in from the laptop that runs Pure Data. The USB cable powers the UNO and at the same time emits processed data to the laptop through the serial bus.

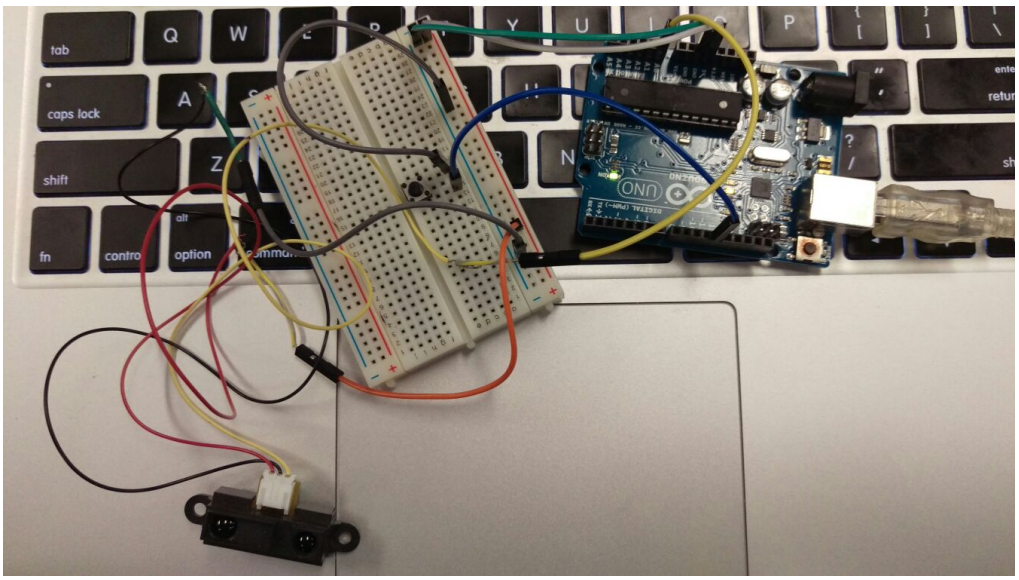

Figure 3: Arduino UNO Circuit

Hence, the circuit emits processed analog distance readings and the octave range reading from the button simultaneously over the USB port to Pure Data. There is no extra power required to power the IR sensor other than the 5V Uno power supply.

5 Implementation

The IR Sensor and button can be easily setup with the Arduino Uno. We will setup a delay of 5ms in the loop() function of the Arduino sketch to read the IR sensor reading. The ADC converted input voltages are often irregular and will need a high pass, low pass filter to cut out the noise in the signal. This is done by using a simple hard bound for low pass and high pass. The Arduino also makes sure every 20 readings are averaged to smooth out the signal before sending the message to Pure Data.

The button functions just like a regular on/off button. The on/off instead cycles through a range from 1 to 3. The range is another global message passed to the Pure Data to facilitate octave switches in a cycle.

6 Modelling on UPPAAL

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays etc.).

The tool has been developed in collaboration between the Design and Analysis of Real-Time Systems group at Uppsala University, Sweden and Basic Research in Computer Science at Aalborg University, Denmark.

The theremin real time system has 3 main parts, IR sensor, Arduino Uno and Button. So we will model the system by dividing it into 3 main templates. Note: The Pure Data does not form a separate template. It simply ends up as the last location/state in the Uno template.

1. IR Sensor : The IR Sensor has to continuously get the reading for every 5ms. So there is a clock x that has a guard of 5ms. Every time this transition is taken, the channel analogRead! synchronizes with the Arduino and sets the clock x to 0. Also

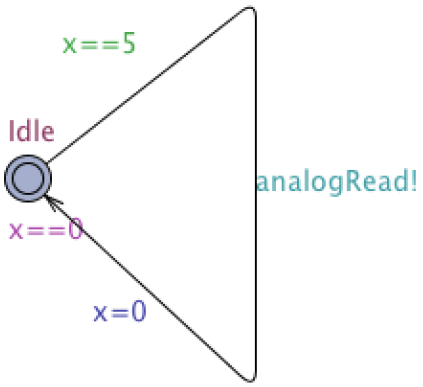

Figure 4: UPPAAL Template: IR Sensor

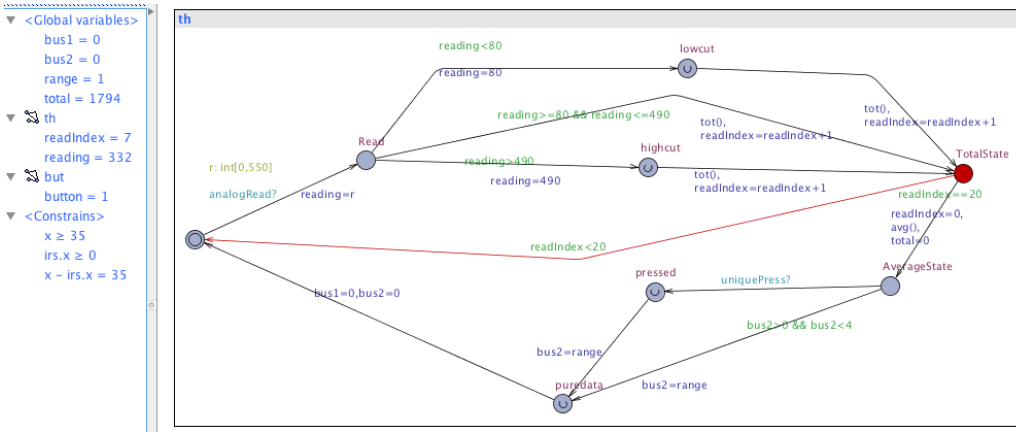

Figure 5: UPPAAL Template: Arduino

there's an invariant for the Idle state. In Figure 4, the IR Sensor has just one location/state called Idle.

2. Uno : In the Arduino template, the channel analogRead synchronizes with IRSensor to get reading and performs a high/low pass filter to cut off few unwanted frequencies. The TotalState adds the reading to a local variable total. Once the readIndex hits 20, the 20 readings added are averaged and that reading is sent out to Pure Data.

There are 2 global variables bus1, bus2 available to all the templates. The IR Sensor fills bus1 with average and bus2 is filled by the Button template.

3. Button :

The button template has an initial Idle and a bpress state. When a button is pressed the range variable is incremented or set equal to 1 if it's previously equal to 3. Also on this transition, the button synchronizes with the Arduino on the uniquePress! channel. If the button is not pressed it will go back to the Idle state and cycle continuously.

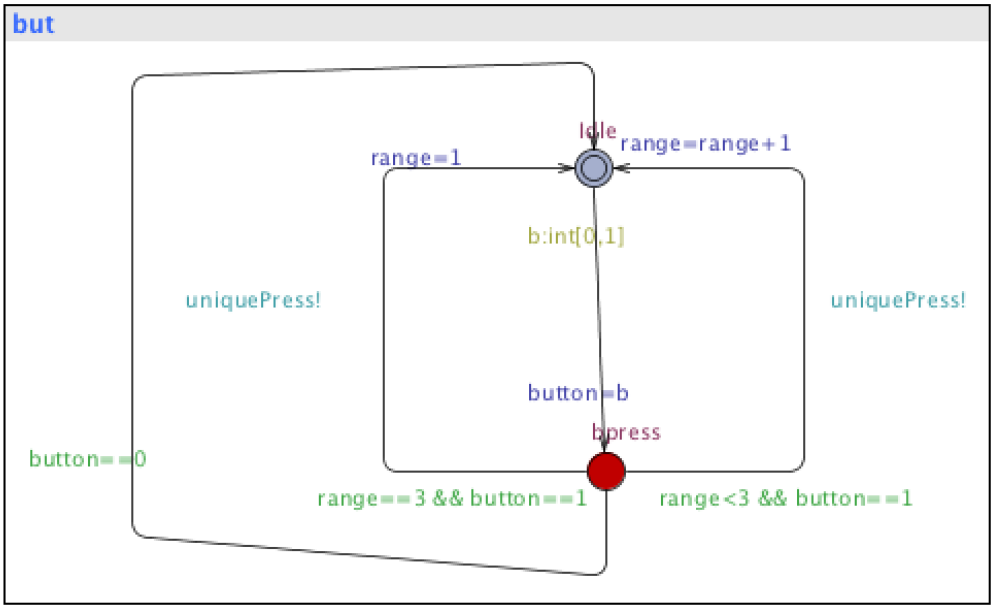

Figure 6: UPPAAL Template: Button

7 Future Work and Conclusion

1. The theremin system currently controls only pitch of the instrument. However, another sensor can be added to control the volume of the instrument simultaneously.
2. Lag and crackling sounds can be avoided by having a smoother note transition system, like a Gaussian smoothing algorithm.
3. It is very easy to manipulate the frequencies to form scales such as major, minor and mixolydian scale. And toggle between each scale while performing.
4. Pure Data also allows a variety of DSP functions like tremolos, sustains and reverbs that can change the feel and texture of the sound.

Conclusion : This is a design prototype of an instrument that was built to setup and understand the work flow. It was modelled in UPPAAL to check for deadlock and unwanted behavior while execution. The model runs continuosly without reaching a deadlock stage or unexpectedly terminating.
It was a great project to understand and implement model checking and at the same time apply DSP, Arduino processing to music development. Ofcourse, there is potential to improve it in numerous ways to build a sturdy and exciting Digital Synth Theremin instrument that can be played without touching it.

8 References

1. Theremin : `https://en.wikipedia.org/wiki/Theremin`
2. Pure Data : `https://puredata.info/`
3. UPPAAL : `http://www.uppaal.org/`
4. Arduino Doc : `https://create.arduino.cc/projecthub`
5. Sharp IR Sensor :`http://www.sharp.co.jp/products/device/doc/opto/gp2y0a21yk_e.pdf`