SEVENTH FRAMEWORK PROGRAMME

AsTeRICS

Arduino Integration and Usage as Sip/Puff Sensor

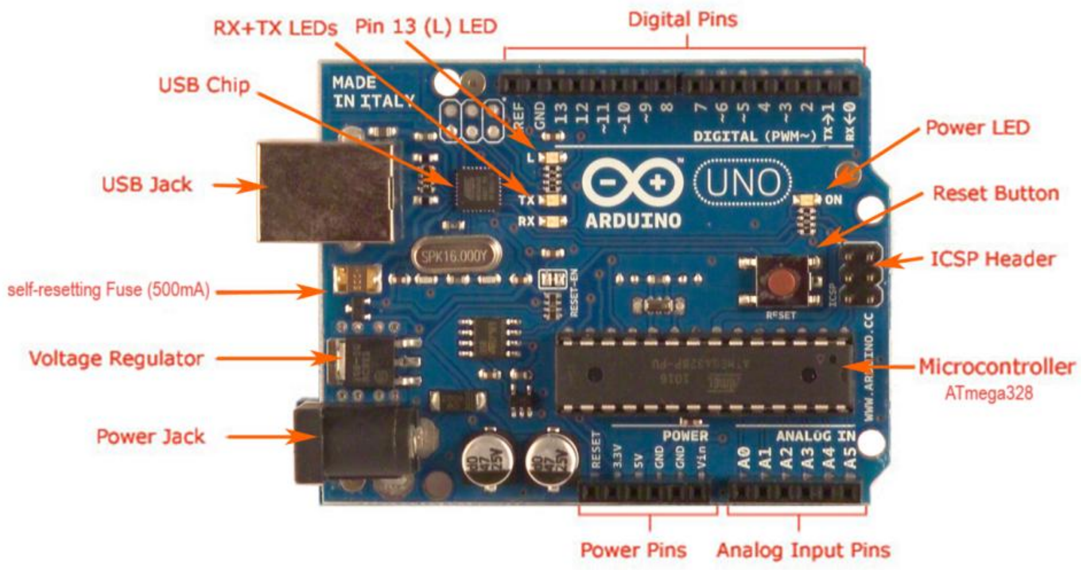
DIY Assembly Guide
Version 1.0 beta

AsTeRICS

European Commission
Information Society and Media

1 Introduction

As described in section 6 of the AsTeRICS User Manual, dedicated AsTeRICS hardware modules are available to interface analog and digital sensors and control external appliances. Alternatively, the popular and affordable “Arduino Uno” microcontroller board can be used to interface the AsTeRICS system to sensors and actuators:

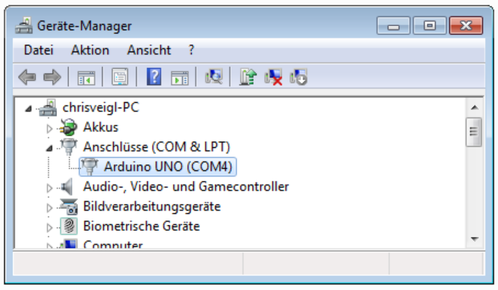


This DIY-guide describes how to set up the Arduino to work with the AsTeRICS system and how to connect a pressure sensor (which could serve as a Sip/Puff input device) to the ARE. Please note that some experience with hardware and microcontroller applications is required for this tutorial. For the commercially available AsTeRICS hardware modules, such background knowledge will not be necessary.

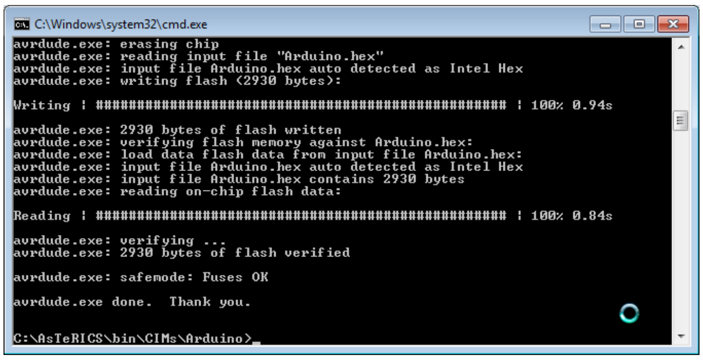
2 Setting up the Arduino

To make the Arduino useable with the AsTeRICS system, a special firmware has to be uploaded into the microcontroller which "talks" to the Arduino ARE plugin. This firmware is contained in the "AsTeRICS-Runtime" download package and can be uploaded with the "Avrdude.exe" tool as follows:

- Connect the "Arduino Uno" to an USB port
- If connected for the first time, the "Arduino UNO.inf" file located in the "CIMS\Arduino" folder has to be selected as a driver for the Arduino hardware
- After that, a COM-Port should appear in the Windows device manager as soon as the Arduino is plugged in. Note the COM-Port number (COM4 in the example):



- Now open a command shell window (e.g. by typing "cmd" <enter> in the windows start menu), and use the "cd" - command to change to the "CIMs/Arduino" folder.
- Type "flash COM4" – where "4" is replaced by the com port number you have got
- Press the reset-button and release it in the moment when you press enter on the keyboard.
- If the firmware upload worked, you should see something like in the following screenshot



```
avrdude.exe: erasing chip
avrdude.exe: reading input file "Arduino.hex"
avrdude.exe: input file Arduino.hex auto detected as Intel Hex
avrdude.exe: writing flash (2930 bytes):

Writing | ################################################## | 100% 0.94s

avrdude.exe: 2930 bytes of flash written
avrdude.exe: verifying flash memory against Arduino.hex:
avrdude.exe: load data flash data from input file Arduino.hex:
avrdude.exe: input file Arduino.hex auto detected as Intel Hex
avrdude.exe: input file Arduino.hex contains 2930 bytes
avrdude.exe: reading on-chip flash data:

Reading | ################################################## | 100% 0.84s

avrdude.exe: verifying ...
avrdude.exe: 2930 bytes of flash verified

avrdude.exe: safemode: Fuses OK

avrdude.exe done.  Thank you.


C:\AsTeRICS\bin\CIMs\Arduino>_
```

The Arduino can now communicate with the Arduino plugin of the ARE. You can connect switches or external devices to the Digital In/Out pins (2-13) and sensors which put out analog voltages from 0-5V to the analog input (A0-A5) of the Arduino. Switches connected to digital input pins can create events triggers, digital output voltages can be changes by incoming events, and the analog sensor values of the pins A0 – A5 are available at the output ports of the plugin.

3 Connecting a Sip/Puff sensor to the Arduino

In the following example, a Sip/Puff Sensor based upon a Freescale MPXV7007G piezoresistive silicon pressure sensor is presented. This sensor delivers an accurate analog output signal in the range of 0 - 5 volts that is proportional to the applied pressure.

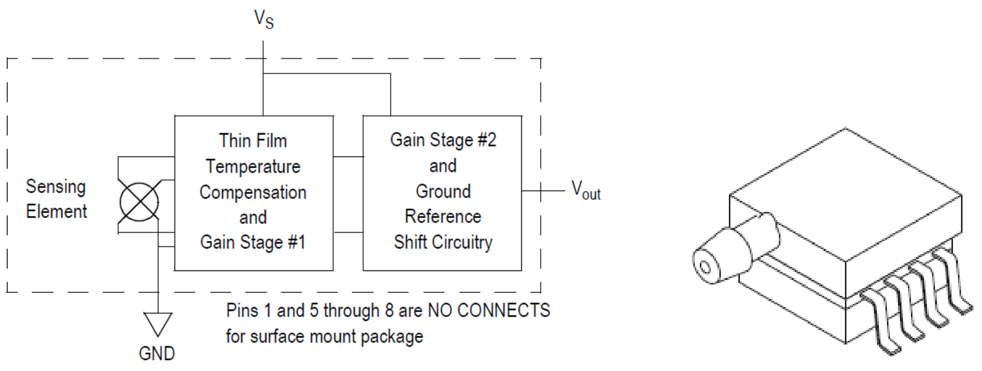
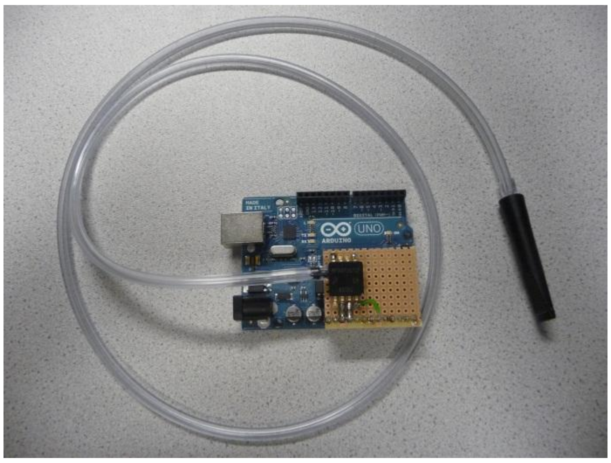

V_S
Sensing Element
Thin Film Temperature Compensation and Gain Stage #1
Gain Stage #2 and Ground Reference Shift Circuitry
V_{out}
GND
Pins 1 and 5 through 8 are NO CONNECTS for surface mount package

Figure 1: Block Diagram and Schematics of the Freescale Pressure Sensor

The utilized model comprises measurement capabilities of positive and negative relative pressures in the range of -7 up to +7 kPa which is very well suited for the operation as a sip/puff sensor. The measurable accuracy is sufficient even for people with very limited lung function and/or sip/puff strength.



The operating voltage of the sensor is 5V, which can be delivered by the Arduino board (+5V pin). Only 3 electrical connections to the sensor are needed: +5V (supply voltage) to Pin 2 of the sensor, GND to Pin 3, and the Voltage output (Pin 3 of the sensor) connected to the Arduino A0 analog input pin. The sensor could be soldered on a breadboard and connected to the Arduino with a pin-header. As tubing system, a PVC tube with an inner diameter of 4mm and a length of approximately 1,5 meters could be used. The above picture shows an Arduino with attached Sip/Puff sensor, a cigarette-tip was used as a mouth piece. This setup could be boxed in a plastic or aluminium case. Please note that this solution is not certified as a product, and has to be used at own risk.

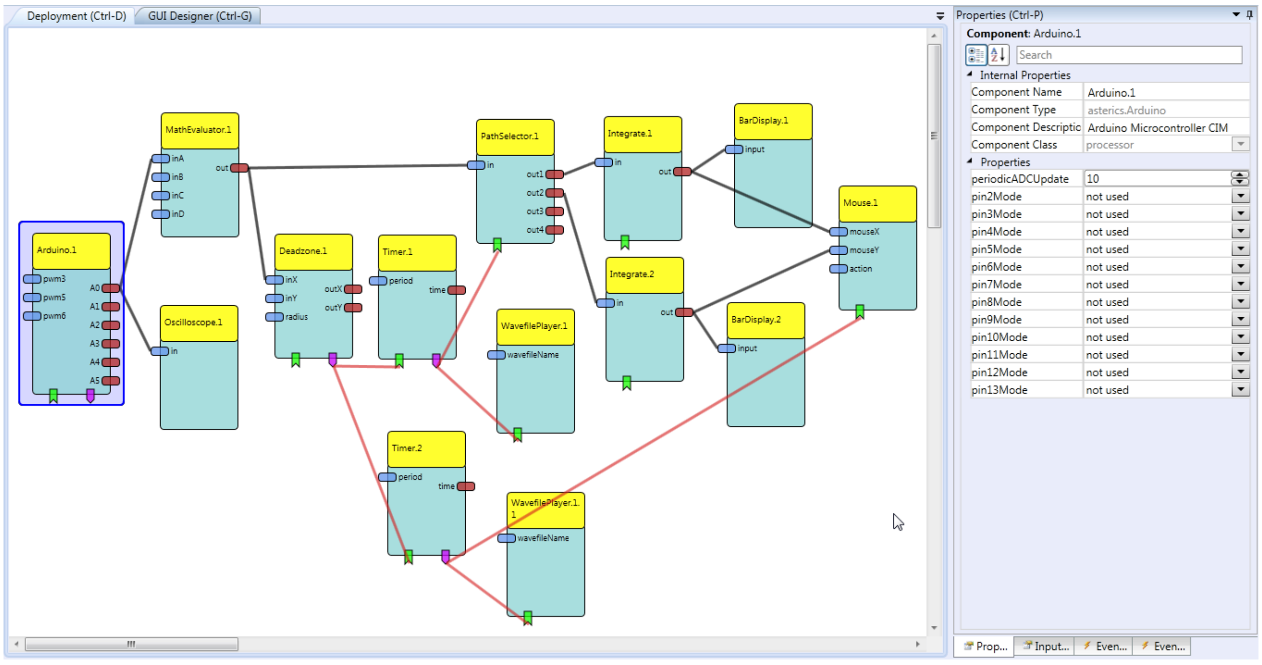
The commercially available AsTeRICS sip/puff sensor will contain a washable mouthpiece with a membrane, which prevents bacteria from entering the sensor.

4 A model for Mouse Control via Sip/Puff Sensor

The following model example uses the analog values of the Sip/Puff sensor to control the mouse in both directions (one after the other). The x-/ y-direction is switched by a certain time of inactivity, left clicks are performed by a longer time of inactivity (dwelling). These events are indicated with acoustic feedback.

- Create a new model and add the "Arduino" plugin
  (Components → Processors → Microcontroller Interface)
- Set the "periodicADCUpdate" property to 10 Milliseconds, this gives 100 updates per second
- Connect the Analog output "A0" to an oscilloscope and test the model (Upload, start).
  The Oscilloscope should show the readings of the pressure sensor. Remember the value when no pressure (no sip or puff) is applied to the sensor
- Add a "MathEvaluator" component and connect "A0" to "inA"
- Set the expression to "(a-508) / 50". (Replace "508" with the sensor value you got for no applied pressure). This creates positive values when blowing into the sensor, and negative values for sipping. The division factor of 50 is used to slow down the mouse speed.
- Add a "Deadzone" component and set the radius to "0.1" – which defines the threshold for inactivity
- Add two "Timer" components – the first will be used to switch X/Y and the second will be used to create a left click.
- Connect the Event Triggers of the "Deadzone" plugin to the event listener ports of both "Timer" plugins. Assign the event "enterZone" to "start" and the event "exitZone" to "stop" and "reset" – please refer to the other model guides in the AsTeRICS user manual if this is unclear.
- Set the "timePeriod" of Timer1 to 1000 milliseconds and the "timePeriod" of Timer2 to 2000
- Add two "WavefilePlayer" plugins (Components → Actuators → Audio and Voice) and enter a desired .wav file name into the "filename" properties e.g. "data/sounds/1.wav" and "data/sounds/2.wav". Please note that these files have to be present in the ARE folder.
- Connect the event trigger port of Timer1 to the event listener port of WavefilePlayer1 and assign "periodFinished" to "play". Do the same with Timer2 and WavefilePlayer2.
- Add a PathSelector plugin (Components → Processors → SignalPathways).
- Connect the output of the MathEvaluator to the input of the PathSelector

- Connect the event trigger port of Timer1 to the event listener port of the PathSelector and assign “periodFinished” to “selectNext”. Thus, the PathSelector component will route the pressure sensor values either to the x or to the y direction of the mouse, depending on the events sent by Timer 1. Note that we did not change the default number of active paths (2) in the properties of the PathSelector as this fits our purpose.
- Add two “Integrate” components (Components → Processors → BasicMath). These will calculate the current x/y positions based upon the relative changes they receive from the pressure sensor. Set the “lowerLimit” of both integrators to “0”, the upper limit of Integrator 1 to the x resolution of your screen, and the upper limit of Integrator 2 to the y resolution of your screen – thereby defining the maximum area for mouse movement.
- Connect “out1” of the PathSelector to Integrate1 and “out2” to Integrate2.
- Add a mouse component and connect the output of Integrate1 to “mouseX” and the output of Integrate2 to “mouseY”
- Optionally, add two “Bardisplay” plugins to show the current x/y coordinates (Components → Actuators → Graphical User Interface).
  Do not forget to set their positions in the GUI designer window.
- Finally, connect the event trigger “periodFinished” of Timer2 to the event listener “leftClick” of the mouse.
-



Properties (Ctrl-P)

Component: Arduino.1

Search

Internal Properties

Component Name	Arduino.1
Component Type	asterics.Arduino
Component Descriptic	Arduino Microcontroller CIM
Component Class	processor

Properties

periodicADCUpdate	10
pin2Mode	not used
pin3Mode	not used
pin4Mode	not used
pin5Mode	not used
pin6Mode	not used
pin7Mode	not used
pin8Mode	not used
pin9Mode	not used
pin10Mode	not used
pin11Mode	not used
pin12Mode	not used
pin13Mode	not used