Using the electret mic to receive audio input on the Beaglebone and detecting a clap

This guide will show you how to connect the Electret Microphone Amplifier-MAX4466 to the Beaglebone and receive audio input through it, and detect a clap close to the elecret amp.

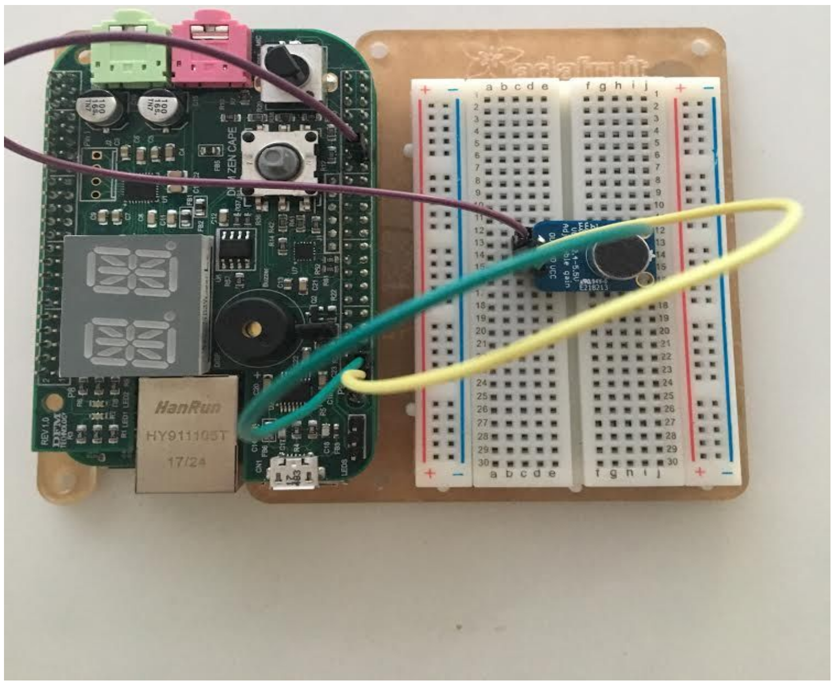
Wiring and Set Up

This guide assumes you have the Zen Cape loaded. To use the amplifier you need to connect the GDN pin to ground, and the VCC to 2.4-5VDC. The OUT pin can then be directly connected to the Beaglebone ADC pin.

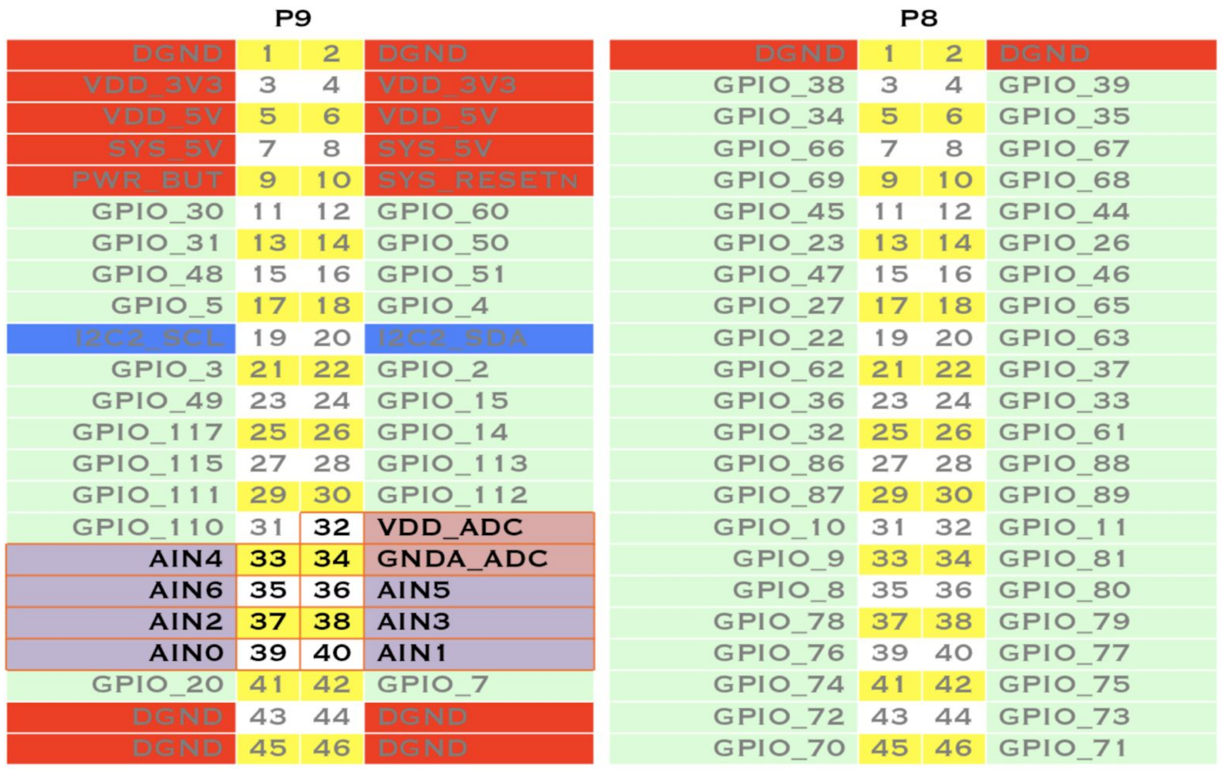
What you need to set up:

- three male to female jumper wires
- Beaglebone
- Electret Microphone Amplifier
- Breadboard

1. Connect the VDC pin to pin 3 on the P9 header of the Zen Cape (VDD_3V3)
2. Connect the GND pin to pin 1 on P9 header of the Zen Cape (DGND)
3. Connect the OUT pin to pin 33 on the P9 header of the Zen Cape (AIN4)



The OUT pin can actually be connected to most ADC pins on the Zen Cape, and it should work but for the example code we read the input on the AIN4 pin. Note that the potentiometer uses the AIN0 pin, and so it is not available to be used by the elecret amp.You might find the diagram below useful. It illustrates where the AIN pins can be found on the Zen cape - they are highlighted in purple.



P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

Once you have everything set up, you need to enable the A2D functionality with the following command:

```
# echo BB-ADC > /sys/devices/platform/bone_capemgr/slots
```

After this step you sometimes need to wait a few minutes, so don't panic. Afterwards you will be able to access the ADC input with the following command:

```
#cd /sys/bus/iio/devices/iio\:device0
```

You can now read the raw analogue input with cat in_voltage4_raw. The highlighted number depends on which AIN pin you connected the mic's OUT pin to.

The process of “cat” ing the input is pretty inefficient so let’s write some code to continuously read and log the input. The code is taken from Dr. Brian Fraser’s A2D guide.

Reading the input with C

First you need to define the path to the A2D input.

```
#define A2D_FILE_VOLTAGE0 "/sys/bus/iio/devices/iio:device0/in_voltage4_raw"
```

The following function reads the input and returns the raw reading.

```
static int getVoltageReading()
{
    FILE *f = fopen(A2D_FILE_VOLTAGE4, "r");
    if (!f)
    {
        printf("ERROR: Unable to open voltage input file. Cape loaded?\n");
        printf("try:   echo BB-ADC > /sys/devices/platform/bone_capemgr/slots\n");
        exit(-1);
    }

    int a2dReading = 0;
    int itemsRead = fscanf(f, "%d", &a2dReading);
    if (itemsRead <= 0)
    {
        printf("ERROR: Unable to read values from voltage input file.\n");
        exit(-1);
    }
    fclose(f);
    return a2dReading;
}
```

You can now continuously loop through the getVoltageReading function, and print the output.

```
while(true)
{

  int A2DReading = getVoltageReading();
  printf("%d", A2DReading);

}
```

Clap Detection

We are going to go through a simple algorithm you can use to detect a clap using the Electret amp wired like we did above. The algorithm was taken from the a research paper and converted to C.

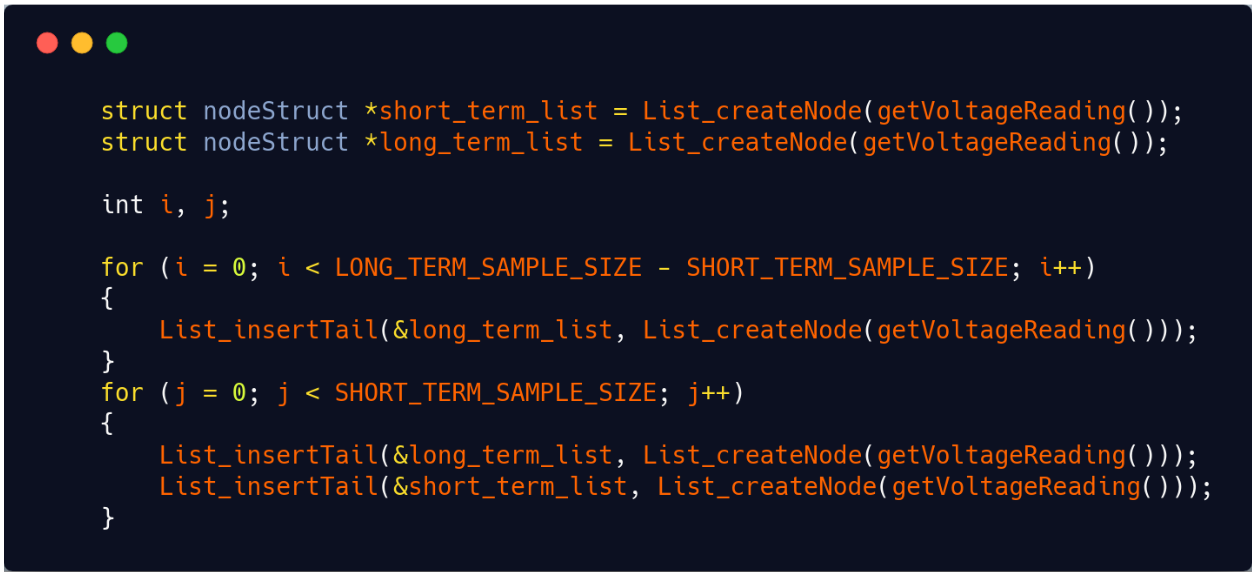
The gist of the algorithm is that you have two buffers that store samples. The first buffer (long term buffer) contains the last 400 analogue readings, and the second (short term buffer) contains the last 20. These buffers are continuously updated in a loop.

The buffers follow the FIFO principle - when we get a new reading, the earliest one is discarded. The buffers can then be easily implemented using a linked list. Each time we read the analogue input, we can remove the tail from the linked list, and add a new node to the head with the latest reading.

When a person claps, the mean of the short term buffer (20 samples) increases, and so we know there was a clap. Sometimes, the readings go down before they go up when a person claps, but overall the mean increases. Knowing this we can define the likelihood of a clap to be the short term buffer mean divided by the long term buffer mean. When this likelihood exceeds a threshold, we know there was a clap.

The following code demonstrates how the algorithm was implemented in C.

We first define our linked lists and fill the short term and long term buffers with the A2D readings.

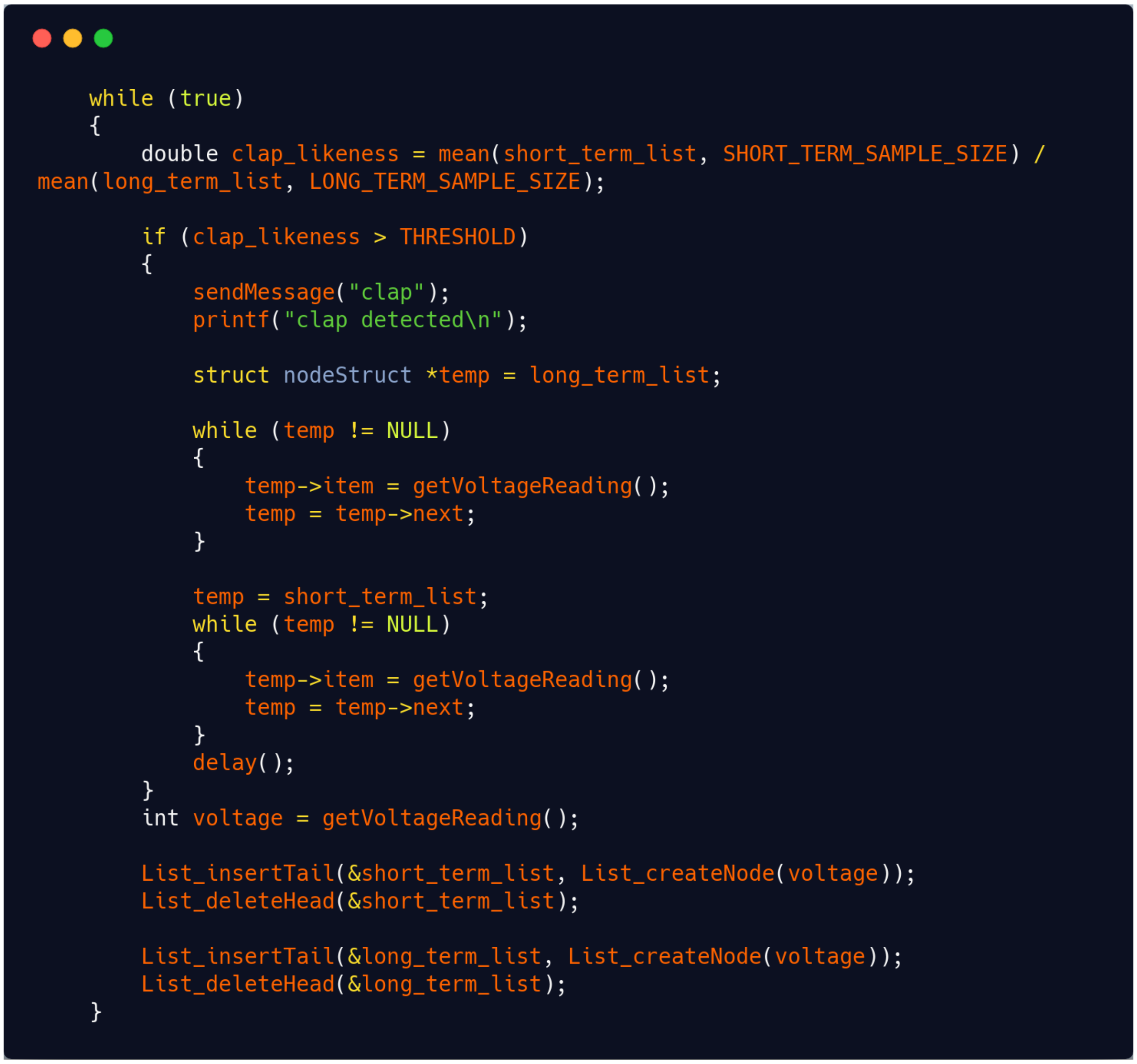


```c
struct nodeStruct *short_term_list = List_createNode(getVoltageReading());
struct nodeStruct *long_term_list = List_createNode(getVoltageReading());

int i, j;

for (i = 0; i < LONG_TERM_SAMPLE_SIZE - SHORT_TERM_SAMPLE_SIZE; i++)
{
    List_insertTail(&long_term_list, List_createNode(getVoltageReading()));
}
for (j = 0; j < SHORT_TERM_SAMPLE_SIZE; j++)
{
    List_insertTail(&long_term_list, List_createNode(getVoltageReading()));
    List_insertTail(&short_term_list, List_createNode(getVoltageReading()));
}
```

Now we can create a loop, and continuously update the long term and short term buffers.



```
    while (true)
    {
        double clap_likeness = mean(short_term_list, SHORT_TERM_SAMPLE_SIZE) /
mean(long_term_list, LONG_TERM_SAMPLE_SIZE);

        if (clap_likeness > THRESHOLD)
        {
            sendMessage("clap");
            printf("clap detected\n");

            struct nodeStruct *temp = long_term_list;

            while (temp != NULL)
            {
                temp->item = getVoltageReading();
                temp = temp->next;
            }

            temp = short_term_list;
            while (temp != NULL)
            {
                temp->item = getVoltageReading();
                temp = temp->next;
            }
            delay();
        }
        int voltage = getVoltageReading();

        List_insertTail(&short_term_list, List_createNode(voltage));
        List_deleteHead(&short_term_list);

        List_insertTail(&long_term_list, List_createNode(voltage));
        List_deleteHead(&long_term_list);
    }
```

Note that after we detect a clap, we refill the long term and short term buffers with new readings. This is because after a clap, there is still an echo from the clap for a few seconds after we have clapped, and so we need to adjust the buffers to depict that the sound in the room is louder.

Troubleshooting

If you are not able to detect a clap, or there are too many false positives, you can try to increase or decrease the threshold.

You may have connected the OUT pin from the amplifier to the wrong analogue pin. You might still be getting a reading, but not the correct one from the amplifier. Make sure the analogue input file you are reading corresponds to the pin connected to the electret amp.