#define arraySize 200

int index = 0;

int IndexofC1;

int n=0; //The number of I\_peaks read

int Peaks[arraySize];

int currentValue[200];

int currentPeakIndex = 0;

int check=1; //To see the jump value once

struct TemplateLibrary

{

int JumpMagnitude;

int ShapeMagnitude;

int ShapeIndex;

int Tolerancec2;

float AvgSteadyState;

float Tolerancec3;

int SettlingTime;

int flag;

};

TemplateLibrary device[10];

int findCurrentPeak ( int currentArray[200] )

{

int currentPeakIndex = 0;

for ( int i = 0; i < arraySize; i++ )

{

if ( currentArray[i] > currentArray[currentPeakIndex] )

{

currentPeakIndex = i;

}

}

return currentPeakIndex;

}

void finding\_currentindex()

{

while ( n < arraySize )

{

for ( int i = 0; i < 200; i++ )

{

currentValue[i] = analogRead(A1);

}

currentPeakIndex = findCurrentPeak ( currentValue );

Peaks[n]=currentValue[currentPeakIndex];

if(Peaks[n]>0 && check==1)

{

IndexofC1=n+1;

check=0;

}

n=n+1;

}

}

void Training() // Function to get the Magnitude and Index of C1

{

finding\_currentindex();

device[index].JumpMagnitude = Peaks[IndexofC1] ;

device[index].flag = 0;

}

void setup()

{

void Training();

}

void loop()

{}