#define arraySize 200

#define sensorpin A1

struct TemplateLibrary

{

int JumpMagnitude;

int index;

int ShapeMagnitude;

int ShapeIndex;

int Tolerancec2;

float AvgSteadyState;

float Tolerancec3;

int SettlingTime;

int flag;

};

struct Runtime

{

int PreviousSteadyState;

int JumpMagnitude;

int ShapeMagnitude;

int ShapeIndex;

float AvgSteadyState;

int SettlingTime;

int Indexofc1;

};

Runtime D;

TemplateLibrary device[10];

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

int findCurrentPeak ( int currentArray[200] );

void finding\_currentindex();

void Training();

void findSettlingTime();

void findTolerance();

void findAverage();

void Display();

void setup()

{

Serial.begin(9600);

analogReference('EXTERNAL');

}

void loop()

{

while(Serial.available()==0);

Serial.read();

Training();

index++;

/\*for(index = 0;index < 10;index ++)

{

while(analogRead(A2)<=0.05);

void Training();

} \*/

Display();

}

//Viki - Code C1

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()

{

n=0;

while ( n < arraySize )

{

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

{

currentValue[i] = analogRead(sensorpin);

// Serial.println(currentValue[i]);

}

currentPeakIndex = findCurrentPeak ( currentValue );

//Serial.println(currentValue[currentPeakIndex]);

Peaks[n]=currentValue[currentPeakIndex];

// Serial.println(Peaks[n]);

n=n+1;

}

int j;

for( j=0; j<arraySize;j++)

{

/\*if( Peaks[j] > 513 && check==1)

{

IndexofC1 = j-1;

check=0;

//break;

}\*/

if(Peaks[j]<=514)

continue;

else

{

Serial.println("inside else");

Serial.println(j);

break;

}

}

device[index].index=j;

D.JumpMagnitude=Peaks[j];

D.Indexofc1=j;

}

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

{

finding\_currentindex();

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

device[index].flag = 0;

// device[index].index=IndexofC1;

// D.JumpMagnitude = Peaks[IndexofC1];

//D.Indexofc1 = IndexofC1;

findShapeMagnitude();

findSettlingTime();

findTolerance();

findAverage();

}

// Hs code - C2

void findShapeMagnitude() {

int indexlocal=D.Indexofc1-1; // local variable for indicating the point of implementation of C2

int i;

int newMagnitude=D.JumpMagnitude; //local variable for succesive magnitude comparision

/\* for( i=indexlocal+1;i<=arraySize;i++)

{

if (Peaks[i]>newMagnitude) //condition for finding maxima

{

newMagnitude=Peaks[i];

}

else

{

break;

}

}

\*/

int fl=1; int cl=1;

for(i=indexlocal; i<arraySize-1; i++)

{

if(Peaks[i]<Peaks[i+1])

continue;

else if(fl==1)

{

fl=0;

for(int k=i;k<arraySize-1;k++)

Peaks[k]=Peaks[k+1];

break;

}

}

for(i=indexlocal; i<arraySize-1; i++)

{

if(Peaks[i]<Peaks[i+1])

continue;

else

{

device[index].ShapeIndex=i-(indexlocal);

device[index].ShapeMagnitude=Peaks[i];

break;

}

}

//device[index].ShapeMagnitude=newMagnitude; // returning the peak

//device[index].ShapeIndex=i; //returning index of peak

}

// Aks COde - C3

void findSettlingTime () // Find settling time

{

Serial.println("entered settiling time");

int localindex=(D.Indexofc1)+device[index].ShapeIndex-1;

int count = 0; // Skipping the first occurance of a positive rise

int flag = 0; // To break the cycle

Serial.print("Local Index:");

Serial.println(localindex);

for ( int i = localindex; i < (arraySize - 1); i++ )

{

if (( Peaks[i] <= Peaks[i+1] ) && ( count == 0 ))

{

count = 1; // First positive rise

continue;

}

if (( Peaks[i] <= Peaks[i+1] ) && (flag == 0 )) // Second rise

{

device[index].SettlingTime = i-localindex;

flag = 1;

break;

}

}

}

void findTolerance ()

{

int toleranceArray[10]; // To store the discreet values

int localindex=device[index].SettlingTime+((D.Indexofc1)+device[index].ShapeIndex-1);

int count = 0; // Number of steady state values

int flag = 0;// To check if the value has occured before

int temp; //For swapping

float average = 0;// average for tolerance

for ( int i = 0; i < 10; i++ )

{

toleranceArray[i] = 0;

}

for ( int i = localindex; i < arraySize; i ++ )// To store only unique values

{

flag = 0;

for ( int j = 0; j < 10; j++ )

{

if ( Peaks[i] == toleranceArray[j] )// The value is already there in the toleranceArray

{

flag = 1;

}

}

if ( flag == 0 )// The value is not in the toleranceArray

{

toleranceArray[count] = Peaks[i];//Add value to toleranceArray

count = count + 1;//Increase last index in tolerance array

}

}

for ( int i = 0; i < count; i++ )// Sort the toleranceArray using bubble sort

{

for ( int j = 0; j < count-i-1; j++ )

{

if ( toleranceArray[i] > toleranceArray[j] )

{

temp = toleranceArray [i];

toleranceArray[i] = toleranceArray[j];

toleranceArray[j] = temp;

}

}

}

for ( int i = 0; i < count; i++ ) // Find average

{

average = average + toleranceArray[i];

}

average = average / count;

device[index].Tolerancec3 = ( abs ( toleranceArray[0] - toleranceArray[count] ) / average ); //Tolerance

}

void findAverage ()

{

float average = 0;// To find average of steady state

int localindex=device[index].SettlingTime+((D.Indexofc1)+device[index].ShapeIndex-1);

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

{

average = average + Peaks[i];

}

Serial.print("Average:");

Serial.println(average);

device[index].AvgSteadyState = average / ( arraySize - localindex ); // To find steady state average from settling time to end

}

void Display()

{

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

{

Serial.println(Peaks[j]);

}

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

{

Serial.print("Device ");

Serial.println(i+1);

Serial.print("index of c1");

Serial.println(device[i].index);

Serial.print("Jump Magnitude:");

Serial.println(device[i].JumpMagnitude);

Serial.print("ShapeMagnitude:");

Serial.println(device[i].ShapeMagnitude);

Serial.print("shapeIndex:");

Serial.println(device[i].ShapeIndex);

Serial.print("Average Steady State:");

Serial.println(device[i].AvgSteadyState);

Serial.print("Settling Time:");

Serial.println(device[i].SettlingTime);

Serial.print("Tolerance c3:");

Serial.println(device[i].Tolerancec3);

}

}