Arduino

```
/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define lines 57
#define word_size 16

int a=1<<8 & 63;
/***MOME MEMORY***/
const char *mem[lines][2]={
  {"65:300","0"},
  {"0:0","65"},
  {"0:16","65"},
  {"0:1","65"},
  {"0:4","65"},
  {"0:17","65"},
  {"0:20","65"},
  {"0:5","65"},
  {"65:28","0"},
  {"65:29","0"},
  {"65:23","0"},
  {"65:31","0"},
  {"65:7","0"},
  {"65:12","0"},
  {"65:13","0"},
  {"65:36","0"},
  {"65:32","0"},
  {"65:33","0"},
  {"65:37","0"},
  {"65:39","0"},
  {"65:40","0"},
  {"65:41","0"},
  {"65:28","0"},
  {"65:24","0"},
  {"65:25","0"},
  {"65:56","0"},
  {"65:57","0"},
  {"65:56","0"},
  {"65:2","0"},
```

```
  {"65:34","0"},
  {"65:35","0"},
  {"65:3","0"},
  {"65:25","0"},
  {"65:53","0"},
  {"65:22","0"},
  {"65:58","0"},
  {"65:59","0"},
  {"65:60","0"},
  {"65:61","0"},
  {"65:54","0"},
  {"65:55","0"},
  {"65:62","0"},
  {"65:63","0"},
  {"65:42","0"},
  {"65:43","0"},
  {"65:44","0"},
  {"65:45","0"},
  {"65:46","0"},
  {"65:47","0"},
  {"65:50","0"},
  {"65:51","0"},
  {"65:52","0"},
  {"65:18","0"},
  {"65:19","0"},
  {"65:9","0"},
  {"65:12","0"},
  {"65:49","0"}
};


//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);
char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
char *intostr(int value);
void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  char keygen[2][word_size];
```

```
char hist[word_size];
int input;
char Input[word_size];
char response[word_size];
int counter;
//inic key
strcpy(keygen[0],"0");//output
strcpy(keygen[1],"0");//input
strcpy(hist,"0");
//mem prepared for depth 2 in FSM (finite state machine).

/********************CICLOS DE MAQUINA**********************/
for(counter=0;TRUE;strcpy(hist,Input)){

  //strcpy(input,63 & PINC);
  input=63 & PINC;
  delay(60);

  strcpy(Input,keygen[0]);
  strcat(Input,":");
  strcat(Input,intostr(input));

  //if(!(Serial.available()>0)){
    //Serial.print(input);
    //Serial.print(" , ");
    //Serial.println(counter);
  //}

  //timer setup
  if(!strcmp(keygen[0],"65")){
    counter++;
  }else{
    counter = 0;
  }
  //catch timer
  if(counter==150){
    strcpy(Input,"65:300");
    counter = 0;
  }//15000
  //end timer

  /**********************/
  if(!strcmp(Input,hist))
    continue;
  //if(!(Serial.available()>0)){
    //Serial.println(input);
  //}

  /****************Search and apply changes****************/
```

```
    strcpy(response,LMOME(mem,keygen,Input));

    PORTB=63 & atoi(response);

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******
int MOME(const int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
******/
//LMOME from matrix
char *LMOME(const char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
```

```
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/***/
char *intostr(int value){
  int i;
  int temp=value;
  char *x;
  x=(char*)calloc(sizeof(char),12);
  for(i=0;temp!=0 && i<12;i++,temp/=10);
    x[i]='\0';i--;
  if(!(i<12))
    return NULL;
  for(temp=value,temp%=10;temp!=0;x[i]=(char)temp+48,value/=10,temp=value,i--,temp%=10);
  //printf("numero:- %s\n",x);
  return x;
}


//& bitwise AND, && logical bool and
/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***MOME MEMORY***/
const int mem[lines][3]={
  {65,300,0},
  {0,0,65},
  {0,16,65},
  {0,1,65},
  {0,4,65},
```

{0,17,65},
{0,20,65},
{0,5,65},
{65,28,0},
{65,29,0},
{65,23,0},
{65,31,0},
{65,7,0},
{65,12,0},
{65,13,0},
{65,36,0},
{65,32,0},
{65,33,0},
{65,37,0},
{65,39,0},
{65,40,0},
{65,41,0},
{65,28,0},
{65,24,0},
{65,25,0},
{65,56,0},
{65,57,0},
{65,56,0},
{65,2,0},
{65,34,0},
{65,35,0},
{65,3,0},
{65,25,0},
{65,53,0},
{65,22,0},
{65,58,0},
{65,59,0},
{65,60,0},
{65,61,0},
{65,54,0},
{65,55,0},
{65,62,0},
{65,63,0},
{65,42,0},
{65,43,0},
{65,44,0},
{65,45,0},
{65,46,0},
{65,47,0},
{65,50,0},
{65,51,0},
{65,52,0},
{65,18,0},
{65,19,0},

```
  {65,9,0},
  {65,12,0},
  {65,49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //mem prepared for depth 2 in FSM (finite state machine).

  /*******************CICLOS DE MAQUINA**********************/
  for(counter=0,hist=0;TRUE;hist=input){

    input = 63 & PINC;
    delay(60);

    //if(!(Serial.available()>0)){
      //Serial.print(input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(keygen[0] == 65){
      counter++;
    }else{
      counter = 0;
    }
```

```
    //catch timer
    if(counter == 150){
      input = 300;
      counter = 0;
    }//15000
    //end timer

    /**********************/
    if(input == hist)
      continue;
    //if(!(Serial.available()>0)){
      //Serial.println(input);
    //}

    /****************Search and apply changes****************/

    response=MOME(mem,keygen,input);

    PORTB=63 & response;

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
```

```
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/
/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***MOME MEMORY***/
int mem[lines][3]={
  {65,300,0},
  {0,0,65},
  {0,16,65},
  {0,1,65},
  {0,4,65},
  {0,17,65},
  {0,20,65},
  {0,5,65},
  {65,28,0},
  {65,29,0},
  {65,23,0},
  {65,31,0},
  {65,7,0},
  {65,12,0},
  {65,13,0},
  {65,36,0},
  {65,32,0},
  {65,33,0},
  {65,37,0},
  {65,39,0},
  {65,40,0},
  {65,41,0},
  {65,28,0},
  {65,24,0},
  {65,25,0},
  {65,56,0},
```

```
  {65,57,0},
  {65,56,0},
  {65,2,0},
  {65,34,0},
  {65,35,0},
  {65,3,0},
  {65,25,0},
  {65,53,0},
  {65,22,0},
  {65,58,0},
  {65,59,0},
  {65,60,0},
  {65,61,0},
  {65,54,0},
  {65,55,0},
  {65,62,0},
  {65,63,0},
  {65,42,0},
  {65,43,0},
  {65,44,0},
  {65,45,0},
  {65,46,0},
  {65,47,0},
  {65,50,0},
  {65,51,0},
  {65,52,0},
  {65,18,0},
  {65,19,0},
  {65,9,0},
  {65,12,0},
  {65,49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
```

```
int keygen[2];
int hist;
int input;
int response;
int counter;
//inic key
keygen[0] = 0;//output
keygen[1] = 0;//input
//mem prepared for depth 2 in FSM (finite state machine).

/********************CICLOS DE MAQUINA*********************/
for(counter=0,hist=0;TRUE;hist=input){

  input = 63 & PINC;
  delay(60);

  if(!(Serial.available()>0)){
    Serial.print(input);
    Serial.print(" , ");
    Serial.println(counter);
  }

  //timer setup
  if(keygen[0] == 65){
    counter++;
  }else{
    counter = 0;
  }
  //catch timer
  if(counter == 150){
    input = 300;
    counter = 0;
  }//15000
  //end timer

  /**********************/
  if(input == hist)
    continue;
  if(!(Serial.available()>0)){
    Serial.println(input);
  }

  /****************Search and apply changes****************/

  response=MOME(mem,keygen,input);

  PORTB=63 & response;

  /*******************************************************/
```

```
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/


#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lines 11
#define TRUE 1

/***MOME Memory***/
int mem[lines][3]={
  {0,63,0},
  {0,62,32},
  {0,61,1},
  {1,62,0},
```

```
  {32,63,32},
  {32,62,0},
  {0,63,0},
  {0,61,33},
  {33,63,33},
  {33,62,0},
  {32,60,0}
};

int MOME(int mem[lines][3],int keygen[2],int input);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;
  keygen[1]=0;

  /*********************************************/

  for(hist=0;TRUE;hist=input){

    //for logic
    //keygen[0]=0;//logic
    //PORTS
    input=PINC;
    delay(30);




    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/
```

```
    response=MOME(mem,keygen,input);
    PORTB=response&B00111111;//masked



    /*********************************************************/
  }//for TRUE
}//loop
/***/
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/***/

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
```

```
}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  for(;TRUE;Hist[B]=Entry[B]){
    //Hist[C]=Entry[C],
    //Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    //if(Entry[C] != Hist[C])
      //Serial.println(Entry[C],DEC);
    //delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
```

```
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***MOME MEMORY***/
const int mem[lines][2]={
  {1<<12|1<<6|0,0},
  {0<<6|0,1},
  {0<<6|16,1},
  {0<<6|1,1},
  {0<<6|4,1},
  {0<<6|17,1},
  {0<<6|20,1},
  {0<<6|5,1},
  {1<<6|28,0},
  {1<<6|29,0},
```

```
  {1<<6|23,0},
  {1<<6|31,0},
  {1<<6|7,0},
  {1<<6|12,0},
  {1<<6|13,0},
  {1<<6|36,0},
  {1<<6|32,0},
  {1<<6|33,0},
  {1<<6|37,0},
  {1<<6|39,0},
  {1<<6|40,0},
  {1<<6|41,0},
  {1<<6|28,0},
  {1<<6|24,0},
  {1<<6|25,0},
  {1<<6|56,0},
  {1<<6|57,0},
  {1<<6|56,0},
  {1<<6|2,0},
  {1<<6|34,0},
  {1<<6|35,0},
  {1<<6|3,0},
  {1<<6|25,0},
  {1<<6|53,0},
  {1<<6|22,0},
  {1<<6|58,0},
  {1<<6|59,0},
  {1<<6|60,0},
  {1<<6|61,0},
  {1<<6|54,0},
  {1<<6|55,0},
  {1<<6|62,0},
  {1<<6|63,0},
  {1<<6|42,0},
  {1<<6|43,0},
  {1<<6|44,0},
  {1<<6|45,0},
  {1<<6|46,0},
  {1<<6|47,0},
  {1<<6|50,0},
  {1<<6|51,0},
  {1<<6|52,0},
  {1<<6|18,0},
  {1<<6|19,0},
  {1<<6|9,0},
  {1<<6|12,0},
  {1<<6|49,0}
};
```

```
//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int LMOME(int mem[lines][2],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //mem prepared for depth 2 in FSM (finite state machine).

  /*******************CICLOS DE MAQUINA*********************/
  for(counter=0,hist=0;TRUE;hist=input){

    input = keygen[0]<<6|(63 & PINC);
    delay(60);

    //if(!(Serial.available()>0)){
      //Serial.print(input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(keygen[0] == 1){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter == 150){
      input = 1<<12|1<<6|0;
      counter = 0;
    }//15000
```

```
    //end timer

    /**********************/
    if(input == hist)
      continue;
    /****************Search and apply changes***************/

    if(!(Serial.available()>0)){
      Serial.println(keygen[0]);
    }

    response=MOME(mem,keygen,input);

    PORTB=63 & response;

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
```

```
}
/******/

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***MOME MEMORY***/
const int mem[lines][2]={
  {1<<12|1<<6|0,0},
  {0<<6|0,1},
  {0<<6|16,1},
  {0<<6|1,1},
  {0<<6|4,1},
  {0<<6|17,1},
  {0<<6|20,1},
  {0<<6|5,1},
  {1<<6|28,0},
  {1<<6|29,0},
  {1<<6|23,0},
  {1<<6|31,0},
  {1<<6|7,0},
  {1<<6|12,0},
  {1<<6|13,0},
  {1<<6|36,0},
  {1<<6|32,0},
  {1<<6|33,0},
  {1<<6|37,0},
  {1<<6|39,0},
  {1<<6|40,0},
  {1<<6|41,0},
  {1<<6|28,0},
  {1<<6|24,0},
  {1<<6|25,0},
  {1<<6|56,0},
  {1<<6|57,0},
  {1<<6|56,0},
  {1<<6|2,0},
```

```
  {1<<6|34,0},
  {1<<6|35,0},
  {1<<6|3,0},
  {1<<6|25,0},
  {1<<6|53,0},
  {1<<6|22,0},
  {1<<6|58,0},
  {1<<6|59,0},
  {1<<6|60,0},
  {1<<6|61,0},
  {1<<6|54,0},
  {1<<6|55,0},
  {1<<6|62,0},
  {1<<6|63,0},
  {1<<6|42,0},
  {1<<6|43,0},
  {1<<6|44,0},
  {1<<6|45,0},
  {1<<6|46,0},
  {1<<6|47,0},
  {1<<6|50,0},
  {1<<6|51,0},
  {1<<6|52,0},
  {1<<6|18,0},
  {1<<6|19,0},
  {1<<6|9,0},
  {1<<6|12,0},
  {1<<6|49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int LMOME(int mem[lines][2],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
```

```
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //FSM (finite state machine).

  /*******************CICLOS DE MAQUINA*********************/
  for(counter=0,hist=0;TRUE;hist=input){

    input = keygen[0]<<6|(63 & PINC);//FSM key
    delay(60);

    //if(!(Serial.available()>0)){
      //Serial.print(input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(keygen[0] == 1){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter == 150){
      input = 1<<12|1<<6|0;
      counter = 0;
    }//15000
    //end timer

    /**********************/
    if(input == hist)
      continue;
    /****************Search and apply changes***************/

    //if(!(Serial.available()>0)){
      //Serial.println(keygen[0]);
    //}

    response=MOME(mem,keygen,input);

    PORTB=63 & response;

    /*********************************************************/
  }
}
/******/
```

```
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***MOME MEMORY***/
const int mem[lines][3]={
```

{65,300,0},
{0,0,65},
{0,16,65},
{0,1,65},
{0,4,65},
{0,17,65},
{0,20,65},
{0,5,65},
{65,28,0},
{65,29,0},
{65,23,0},
{65,31,0},
{65,7,0},
{65,12,0},
{65,13,0},
{65,36,0},
{65,32,0},
{65,33,0},
{65,37,0},
{65,39,0},
{65,40,0},
{65,41,0},
{65,28,0},
{65,24,0},
{65,25,0},
{65,56,0},
{65,57,0},
{65,56,0},
{65,2,0},
{65,34,0},
{65,35,0},
{65,3,0},
{65,25,0},
{65,53,0},
{65,22,0},
{65,58,0},
{65,59,0},
{65,60,0},
{65,61,0},
{65,54,0},
{65,55,0},
{65,62,0},
{65,63,0},
{65,42,0},
{65,43,0},
{65,44,0},
{65,45,0},
{65,46,0},
{65,47,0},

```
  {65,50,0},
  {65,51,0},
  {65,52,0},
  {65,18,0},
  {65,19,0},
  {65,9,0},
  {65,12,0},
  {65,49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //mem prepared for depth 2 in FSM (finite state machine).

  /********************CICLOS DE MAQUINA**********************/
  for(counter=0,hist=0;TRUE;hist=input){

    input = 63 & PINC;
    delay(60);

    //if(!(Serial.available()>0)){
      //Serial.print(input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
```

```
    if(keygen[0] == 65){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter == 150){
      input = 300;
      counter = 0;
    }//15000
    //end timer

    /**********************/
    if(input == hist)
      continue;
    //if(!(Serial.available()>0)){
      //Serial.println(input);
    //}

    /****************Search and apply changes****************/

    response=MOME(mem,keygen,input);

    PORTB=63 & response;

    /************************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
```

```
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/
/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define TRUE 1
#define FALSE 0
#define lines 57
#define buf_size 32

/***LMOME MEMORY***/
const char *mem[lines][2]={
  {"temporizador","0"},
  {"0:0","65"},
  {"0:16","65"},
  {"0:1","65"},
  {"0:4","65"},
  {"0:17","65"},
  {"0:20","65"},
  {"0:5","65"},
  {"65:28","0"},
  {"65:29","0"},
  {"65:23","0"},
  {"65:31","0"},
  {"65:7","0"},
  {"65:12","0"},
  {"65:13","0"},
  {"65:36","0"},
  {"65:32","0"},
  {"65:33","0"},
  {"65:37","0"},
```

```
  {"65:39","0"},
  {"65:40","0"},
  {"65:41","0"},
  {"65:28","0"},
  {"65:24","0"},
  {"65:25","0"},
  {"65:56","0"},
  {"65:57","0"},
  {"65:56","0"},
  {"65:2","0"},
  {"65:34","0"},
  {"65:35","0"},
  {"65:3","0"},
  {"65:25","0"},
  {"65:53","0"},
  {"65:22","0"},
  {"65:58","0"},
  {"65:59","0"},
  {"65:60","0"},
  {"65:61","0"},
  {"65:54","0"},
  {"65:55","0"},
  {"65:62","0"},
  {"65:63","0"},
  {"65:42","0"},
  {"65:43","0"},
  {"65:44","0"},
  {"65:45","0"},
  {"65:46","0"},
  {"65:47","0"},
  {"65:50","0"},
  {"65:51","0"},
  {"65:52","0"},
  {"65:18","0"},
  {"65:19","0"},
  {"65:9","0"},
  {"65:12","0"},
  {"65:49","0"}
};


//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);
char *LMOME(char *memory[lines][2],char keygen[2][buf_size],char input[buf_size]);
int intostr(int value,char *x);
void setup()
{
  //start serial port at 9600 bps:
```

```
  Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  char keygen[2][buf_size];
  char hist[buf_size];
  char input[buf_size];
  char Input[buf_size];
  char response[buf_size];
  int counter;
  //inic key
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"0");
  //mem prepared for depth 2 in FSM (finite state machine).

  /*******************CICLOS DE MAQUINA**********************/
  for(counter=0;TRUE;strcpy(hist,Input)){

    //strcpy(input,63 & PINC);
    intostr(63 & PINC,input);
    delay(60);

    //logical key
    strcpy(Input,keygen[0]);
    strcat(Input,":");
    strcat(Input,input);

    //if(!(Serial.available()>0)){
      //Serial.print(Input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(!strcmp(keygen[0],"65")){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter==150){
      strcpy(Input,"temporizador");
      counter = 0;
```

```
    }//15000
    //end timer

    /**********************/
    if(!strcmp(Input,hist))
      continue;


    if(!(Serial.available()>0)){
      Serial.println(input);
    }

    /****************Search and apply changes****************/

    strcpy(response,LMOME(mem,keygen,Input));

    PORTB=63 & atoi(response);

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
```

```
    }
  }//for iterator
  return keygen[0];
}
/******/
//LMOME from matrix
char *LMOME(const char *memory[lines][2],char keygen[2][buf_size],char input[buf_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/***/
int intostr(int value,char *x)
{
  int i;
  int temp=value;
  for(i=0;temp!=0;temp/=10,i++);
  x[i]='\0';
  for(i--,temp=value,temp%=10;!(i<0);x[i]=temp+48,value/=10,temp=value,i--,temp%=10);
  return 0;
}

//& bitwise AND, && logical bool and
//solving problems using complicated
//methods make it more viable and strong.

#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lines 8
#define TRUE 1

/***Mealy Memory***/
int mem[lines][4]={
  {0,0,63,0},
  {63,0,62,32},
```

```
  {62,32,63,32},
  {63,32,62,0},
  {62,0,63,0},
  {63,0,61,33},
  {61,33,63,33},
  {63,33,62,0}
};

int MOORE(int mem[lines][3],int keygen[2],int input);
int MEALY(int mem[lines][4],int keygen[2],int input);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;
  keygen[1]=0;

  /*****************************************/

  for(hist=0;TRUE;hist=input){


    input=PINC;
    delay(30);







    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/
    response=MEALY(mem,keygen,input);

    PORTB=response&B00111111;//masked
```

```
    /*********************************************************/
  }//for TRUE
}//loop
/******/
int MOORE(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[0]==input)//evitar redundancia
    return keygen[1];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[1] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=mem[iterator][2];
      //communicate
      break;
    }
  }//for iterator
  return keygen[1];
}
/******/
int MEALY(int mem[lines][4],int keygen[2],int input)
{
  int iterator;
  int keyfound;//evita redundancia
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==keygen[1] && mem[iterator]
[2]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=mem[iterator][3];
      break;
    }
  }//for iterator
  return keygen[1];
}
/******/

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
```

```
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***MOME MEMORY***/
const int mem[lines][2]={
  {1<<12|1<<6|0,0},
  {0<<6|0,1},
  {0<<6|16,1},
  {0<<6|1,1},
  {0<<6|4,1},
  {0<<6|17,1},
  {0<<6|20,1},
  {0<<6|5,1},
  {1<<6|28,0},
  {1<<6|29,0},
  {1<<6|23,0},
  {1<<6|31,0},
  {1<<6|7,0},
  {1<<6|12,0},
  {1<<6|13,0},
  {1<<6|36,0},
  {1<<6|32,0},
  {1<<6|33,0},
  {1<<6|37,0},
  {1<<6|39,0},
  {1<<6|40,0},
  {1<<6|41,0},
  {1<<6|28,0},
  {1<<6|24,0},
  {1<<6|25,0},
  {1<<6|56,0},
  {1<<6|57,0},
  {1<<6|56,0},
  {1<<6|2,0},
  {1<<6|34,0},
  {1<<6|35,0},
  {1<<6|3,0},
  {1<<6|25,0},
  {1<<6|53,0},
  {1<<6|22,0},
  {1<<6|58,0},
  {1<<6|59,0},
```

```
  {1<<6|60,0},
  {1<<6|61,0},
  {1<<6|54,0},
  {1<<6|55,0},
  {1<<6|62,0},
  {1<<6|63,0},
  {1<<6|42,0},
  {1<<6|43,0},
  {1<<6|44,0},
  {1<<6|45,0},
  {1<<6|46,0},
  {1<<6|47,0},
  {1<<6|50,0},
  {1<<6|51,0},
  {1<<6|52,0},
  {1<<6|18,0},
  {1<<6|19,0},
  {1<<6|9,0},
  {1<<6|12,0},
  {1<<6|49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int LMOME(int mem[lines][2],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //FSM (finite state machine).

  /*******************CICLOS DE MAQUINA*********************/
```

```
  for(counter=0,hist=0;TRUE;hist=input){

    input = keygen[0]<<6|(63 & PINC);//FSM key
    delay(60);

    //if(!(Serial.available()>0)){
      //Serial.print(input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(keygen[0] == 1){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter == 150){
      input = 1<<12|1<<6|0;
      counter = 0;
    }//15000
    //end timer

    /**********************/
    if(input == hist)
      continue;
    /****************Search and apply changes****************/

    //if(!(Serial.available()>0)){
      //Serial.println(keygen[0]);
    //}

    response=LMOME(mem,keygen,input);

    PORTB=63 & response;

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
```

```
      continue;
    flag=0;
    }
  return num;
}
/******/
int LMOME(const int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/

#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lmoore 10
#define lmealy 11
#define TRUE 1

/***Mealy Memory***/
int memmealy[lmealy][4]={
  {0,0,63,0},
  {0,0,61,1},
  {0,0,62,32},
  {0,0,62,32},
  {0,32,63,32},
  {0,32,62,0},
  {0,0,63,0},
  {0,0,61,33},
  {0,33,63,33},
  {0,33,62,0},
  {0,32,60,0}
};

/***Moore Memory***/
int memmoore[lmoore][3]={
```

```
  {0,63,0},
  {0,62,32},
  {0,61,1},
  {32,63,32},
  {32,62,0},
  {0,63,0},
  {0,61,33},
  {33,63,33},
  {33,62,0},
  {32,60,0}
};

int MOORE(int mem[lmoore][3],int keygen[2],int input);
int MEALY(int mem[lmealy][4],int keygen[2],int input);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;
  keygen[1]=0;

  /*******************************************/

  for(hist=0;TRUE;hist=input){

    //for logic
    keygen[0]=0;
    //keygen[1]=0;
    //PORTS
    input=PINC;
    delay(30);
```

```
    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/
    response=MEALY(memmealy,keygen,input);

    //response=MOORE(memmoore,keygen,input);
    PORTB=response&B00111111;//masked



    /*********************************************************/
  }//for TRUE
}//loop
/******/
int MOORE(int mem[lmoore][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[0]==input)//evitar redundancia
    return keygen[1];
  for(iterator=0;iterator<lmoore;iterator++){
    keyfound=(mem[iterator][0]==keygen[1] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=mem[iterator][2];
      //communicate
      break;
    }
  }//for iterator
  return keygen[1];
}
/******/
int MEALY(int mem[lmealy][4],int keygen[2],int input)
{
  int iterator;
  int keyfound;//evita redundancia
  for(iterator=0;iterator<lmealy;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==keygen[1] && mem[iterator]
[2]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=mem[iterator][3];
      break;
    }
  }//for iterator
  return keygen[1];
```

```
}
/******/

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;

  Serial.flush();
  Serial.println(EOF,DEC);
  //INIC
  for( Hist[C]=0,Hist[B]=0; TRUE; Hist[C] = Entry[C], Hist[B] = Entry[B]){
    //ENTRADA Pinos
    Entry[C]=PINC;
    //ENTRADA Serial
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }

    if((Entry[C] == Hist[C]) && (Entry[B] == Hist[B]))
      continue;
```

```
    //if(Entry[C] != Hist[C])
    Serial.println(Entry[C],DEC);
    delay(30);
    PORTB = Entry[B];
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/********/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
```

```
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char* x);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{
  uint8_t i=0;
  uint8_t Entry[2];
  uint8_t Hist[2];
  char IncomingByte;
  char State[BufSize];
  //INIC
  for( Hist[C]=0,Hist[B]=0; TRUE; Hist[C] = Entry[C], Hist[B] = Entry[B]){
    //ENTRADA Pinos
    Entry[C]=PINC;
    //ENTRADA Serial
    if(Serial.available()){
      delay(25);//wait for incoming data.
      for(i = 0; IncomingByte = Serial.read(); i++){
           if((IncomingByte == '\r') || (IncomingByte == '\n')){
             State[i] = '\0';
             Serial.flush();
             break;

           }else{
             State[i]=IncomingByte;

           }

      }
      Entry[B] = getnum(State);
```

```
    }
    if((Entry[C] == Hist[C]) && (Entry[B] == Hist[B]))
      continue;
    //if(Entry[C] != Hist[C])
    Serial.println(Entry[C],DEC);
    PORTB = Entry[B];
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}

//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define line 8
#define TRUE 1

/***Mealy Memory***/
int mem[line][4]={
  {0,0,63,0},
  {63,0,62,32},
  {62,32,63,32},
  {63,32,62,0},
  {62,0,63,0},
  {63,0,61,33},
  {61,33,63,33},
```

```
  {63,33,62,0}
};

int MOORE(int mem[line][3],int keygen[2],int input);
int MEALY(int mem[lines][4],int keygen[2],int input);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[3];
  int hist;
  int response;
  //Inic
  keygen[0]=0;
  keygen[1]=0;

  /*********************************************/

  for(hist=0;TRUE;hist=keygen[2]){


    input=PINC;
    delay(30);







    if(keygen[2]==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/
    response=MEALY(mem,keygen,input);




    /*****************************************************/
  }//for TRUE
}//loop
```

```
/******/
int MOORE(int mem[line][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[0]==input)//evitar redundancia
    return keygen[1];
  for(iterator=0;iterator<line;iterator++){
    keyfound=(mem[iterator][0]==keygen[1] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=mem[iterator][2];
      //communicate
      break;
    }
  }//for iterator
  return keygen[1];
}
/******/
int MEALY(int mem[lines][4],int keygen[2],int input)
{
  int iterator;
  int keyfound;//evita redundancia
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==keygen[1] && mem[iterator]
[2]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=mem[iterator][3];
      break;
    }
  }//for iterator
  return keygen[1];
}
/******/
/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
```

```
#define lines 57

/***MOME MEMORY***/
const int mem[lines][3]={
  {65,300,0},
  {0,0,65},
  {0,16,65},
  {0,1,65},
  {0,4,65},
  {0,17,65},
  {0,20,65},
  {0,5,65},
  {65,28,0},
  {65,29,0},
  {65,23,0},
  {65,31,0},
  {65,7,0},
  {65,12,0},
  {65,13,0},
  {65,36,0},
  {65,32,0},
  {65,33,0},
  {65,37,0},
  {65,39,0},
  {65,40,0},
  {65,41,0},
  {65,28,0},
  {65,24,0},
  {65,25,0},
  {65,56,0},
  {65,57,0},
  {65,56,0},
  {65,2,0},
  {65,34,0},
  {65,35,0},
  {65,3,0},
  {65,25,0},
  {65,53,0},
  {65,22,0},
  {65,58,0},
  {65,59,0},
  {65,60,0},
  {65,61,0},
  {65,54,0},
  {65,55,0},
  {65,62,0},
  {65,63,0},
  {65,42,0},
  {65,43,0},
```

```
  {65,44,0},
  {65,45,0},
  {65,46,0},
  {65,47,0},
  {65,50,0},
  {65,51,0},
  {65,52,0},
  {65,18,0},
  {65,19,0},
  {65,9,0},
  {65,12,0},
  {65,49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //mem prepared for depth 2 in FSM (finite state machine).

  /*********************CICLOS DE MAQUINA**********************/
  for(counter=0,hist=0;TRUE;hist=input){

    input = 63 & PINC;
    delay(60);

    //if(!(Serial.available()>0)){
      //Serial.print(input);
      //Serial.print(" , ");
```

```
      //Serial.println(counter);
    //}

    //timer setup
    if(keygen[0] == 65){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter == 150){
      input = 300;
      counter = 0;
    }//15000
    //end timer

    /**********************/
    if(input == hist)
      continue;
    //if(!(Serial.available()>0)){
      //Serial.println(input);
    //}

    /****************Search and apply changes****************/

    response=MOME(mem,keygen,input);

    PORTB=63 & response;

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][3],int keygen[2],int input)
{
```

```
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/
/*
//LOGIC from Matrix int
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lines 10
#define TRUE 1

/***MOME Memory***/
int mem[lines][3]={
  {0,63,0},
  {0,62,32},
  {0,61,1},
  {32,63,32},
  {32,62,0},
```

```
  {0,63,0},
  {0,61,33},
  {33,63,33},
  {33,62,0},
  {32,60,0}
};

int MOME(int mem[lines][3],int keygen[2],int input);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;
  keygen[1]=0;

  /*********************************************/

  for(hist=0;TRUE;hist=input){

    //for logic
    keygen[0]=0;
    //keygen[1]=0;
    //PORTS
    input=PINC;
    delay(30);







    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/
```

```
    response=MOME(mem,keygen,input);
    PORTB=response&B00111111;//masked



    /*********************************************************/
  }//for TRUE
}//loop
/***/
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/***/

#include<avr/io.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char* x);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B11111111;
  PORTB=B00000000;

}
void loop()
```

```
{
 uint8_t i=0;
 uint8_t Entry[2];
 uint8_t Past[2];
 uint8_t Hist[2];
 uint8_t num;
 char IncomingByte;
 char State[BufSize];
 Past[C]=Hist[C]=0;
 //INIC
 for( Past[C] = Entry[C], Past[B] = Entry[B]; TRUE; Hist[C] = Entry[C], Hist[B] = Entry[B]){
  //ENTRADA Portas
  Entry[C]=PINC;
  //ENTRADA Serial
  if(Serial.available() > 0){
   delay(25);//wait for incoming data.
   for(i = 0; IncomingByte = Serial.read(); i++){
        if((IncomingByte == '\r') || (IncomingByte == '\n')){
          State[i] = '\0';
          Serial.flush();
          break;

        }else{
          State[i]=IncomingByte;

        }

   }
   Entry[B] = getnum(&State[0]);

  }
  if((Entry[C] == Hist[C]) && (Entry[B] == Hist[B]))
   continue;
  for(delay(10) ; TRUE; Past[C] = Entry[C], Past[B] = Entry[B]){
   if((Entry[C] == Past[C]) && (Entry[B] == Past[B]))
        break;
   /**Processing***/
   if(Entry[C] != Past[C]){
        if(!(Serial.available() > 0)){
          //leituras do microcontrolador.
       num=Entry[C];
          Serial.println(num,DEC);
       //Serial.write(&Entry[C],1);

        }

   }
   if(Entry[B] != Past[B]){
         PORTB = Entry[B];
```

```
      }

    }

  }

}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num) != 0){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}

//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
```

```
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }
```

```
}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lmoore 10
#define lmealy 11
#define TRUE 1

/***Mealy Memory***/
int memmealy[lmealy][4]={
  {0,0,63,0},
  {0,0,61,1},
  {0,0,62,32},
  {0,0,62,32},
  {0,32,63,32},
  {0,32,62,0},
  {0,0,63,0},
  {0,0,61,33},
  {0,33,63,33},
  {0,33,62,0},
  {0,32,60,0}
};
```

```
/***Moore Memory***/
int memmoore[lmoore][3]={
  {0,63,0},
  {0,62,32},
  {0,61,1},
  {32,63,32},
  {32,62,0},
  {0,63,0},
  {0,61,33},
  {33,63,33},
  {33,62,0},
  {32,60,0}
};

int MOORE(int mem[lmoore][3],int keygen[2],int input);
int MEALY(int mem[lmealy][4],int keygen[2],int input);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;
  keygen[1]=0;

  /*****************************************/

  for(hist=0;TRUE;hist=input){

    //for logic
    //keygen[0]=0;
    //keygen[1]=0;
    //PORTS
    input=PINC;
    delay(30);
```

```
    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/
    response=MEALY(memmealy,keygen,input);

    //response=MOORE(memmoore,keygen,input);
    PORTB=response&B00111111;//masked



    /*********************************************************/
  }//for TRUE
}//loop
/******/
int MOORE(int mem[lmoore][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[0]==input)//evitar redundancia
    return keygen[1];
  for(iterator=0;iterator<lmoore;iterator++){
    keyfound=(mem[iterator][0]==keygen[1] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=mem[iterator][2];
      //communicate
      break;
    }
  }//for iterator
  return keygen[1];
}
/******/
int MEALY(int mem[lmealy][4],int keygen[2],int input)
{
  int iterator;
  int keyfound;//evita redundancia
  for(iterator=0;iterator<lmealy;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==keygen[1] && mem[iterator]
[2]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=mem[iterator][3];
      break;
    }
```

```
  }//for iterator
  return keygen[1];
}
/******/

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***MOME MEMORY***/
int mem[lines][3]={
  {65,300,0},
  {0,0,65},
  {0,16,65},
  {0,1,65},
  {0,4,65},
  {0,17,65},
  {0,20,65},
  {0,5,65},
  {65,28,0},
  {65,29,0},
  {65,23,0},
  {65,31,0},
  {65,7,0},
  {65,12,0},
  {65,13,0},
  {65,36,0},
  {65,32,0},
  {65,33,0},
  {65,37,0},
  {65,39,0},
  {65,40,0},
  {65,41,0},
  {65,28,0},
  {65,24,0},
  {65,25,0},
  {65,56,0},
  {65,57,0},
  {65,56,0},
```

```
  {65,2,0},
  {65,34,0},
  {65,35,0},
  {65,3,0},
  {65,25,0},
  {65,53,0},
  {65,22,0},
  {65,58,0},
  {65,59,0},
  {65,60,0},
  {65,61,0},
  {65,54,0},
  {65,55,0},
  {65,62,0},
  {65,63,0},
  {65,42,0},
  {65,43,0},
  {65,44,0},
  {65,45,0},
  {65,46,0},
  {65,47,0},
  {65,50,0},
  {65,51,0},
  {65,52,0},
  {65,18,0},
  {65,19,0},
  {65,9,0},
  {65,12,0},
  {65,49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
```

```
  int input;
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //mem prepared for depth 2 in FSM (finite state machine).

  /*******************CICLOS DE MAQUINA*********************/
  for(counter=0,hist=0;TRUE;hist=input){

    input = 63 & PINC;
    delay(60);

    if(!(Serial.available()>0)){
      Serial.print(input);
      Serial.print(" , ");
      Serial.println(counter);
    }

    //timer setup
    if(keygen[0] == 65){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter == 150){
      input = 300;
      counter = 0;
    }//15000
    //end timer

    /***********************/
    if(input == hist)
      continue;
    if(!(Serial.available()>0)){
      Serial.println(input);
    }

    /****************Search and apply changes****************/

    response=MOME(mem,keygen,input);

    PORTB=63 & response;

    /*****************************************************/
  }
}
```

```
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/


//Este programa é aplicado para botoneiras start/stop,
//aplicado em minha casa. Vamos ver duração
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#include<avr/interrupt.h>
#define TRUE 1
#define FALSE 0
#define column 2
#define lines 52

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
```

```
  DDRC = B00000000;
  PORTC = B11111111;
  DDRB = B11111111;
  PORTB = B00000000;
}

void loop()
{
  int i, l, c;
  int data[3];
  int Hist[3];
  int treat;
  int count=0;
  //inic key
  data[0] = 0;
  data[2] = 0;
  //mem prepared for depth 2 in FSM (finite state machine).
  const int mem[(lines+1)][(column+1)]=
  {
    {65,300, 0},//timer off output 0
    { 0,21, 0},//1
    {65,21,65},//2
    { 0,17,65},//3timer on and output 1
    {65,17,65},//4
    { 0,20,65},//5
    {65,20,65},//6
    { 0, 5,65},//7
    {65, 5,65},//8
    {65,29, 0},//9
    { 0,29, 0},//10
    {65,23, 0},//11
    { 0,23, 0},//12
    {65,31, 0},//13
    { 0,31, 0},//14
    {65, 7, 0},//15
    { 0, 7, 0},//16
    {65,13, 0},//17
    { 0,13, 0},//18
    {65,37, 0},//19
    { 0,37, 0},//20
    {65,28, 0},//21
    { 0,28, 0},//22
    {65,25, 0},//23
    { 0,25, 0},//24
    {65,17,65},//25
    { 0,13, 0},//26
    { 0,53, 0},//27
    {65,53, 0},//28
    { 0,53, 0},//29
```

```
  {65,22, 0},//30
  { 0,22, 0},//31
  {65,61, 0},//32
  { 0,61, 0},//33
  {65,55, 0},//34
  { 0,55, 0},//35
  {65,63, 0},//36
  { 0,63, 0},//37
  {65,52, 0},//38
  { 0,52, 0},//39
  {65,19, 0},//40
  { 0,19, 0},//41
  {65, 9, 0},//42
  { 0, 9, 0},//43
  {65,12, 0},//44
  { 0,12, 0},//45
  { 0,24, 0},//46
  { 0,24, 0},//47
  {65,49, 0},//48
  { 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[2] = data[2], count = 0; TRUE; Hist[1] = data[1]){

  treat = 63 & PINC;
  data[1] = treat;

  delay(60);

  //if(!(Serial.available()>0)){
    //Serial.print(data[3]);
    //Serial.print(" , ");
    //Serial.println(count);
  //}

  //timer setup
  if(data[2] == 65){
    count++;
  }else{
    count = 0;
  }
  //catch timer
  if(count == 15000){
    data[1] = 300;
    count = 0;
  }
  //end timer
```

```
    /**********************/
    if(data[1] == Hist[1])
      continue;
    //if(!(Serial.available()>0)){
      //Serial.println(data[1]);
    //}
    /**********************/
    /**********************Search and apply changes**********************/
    for( l = 0, c = 0; l < lines   ; l++){
      if(c >= column){
        l--;
        data[2] = mem[l][c];
        //update
        data[0] = data[2];
        Hist[0] = data[0];
        //send
        PORTB = 63 & data[2];
        break;
      }
      /***/
      for(c=0; c < column; c++){
        if(data[c] == mem[l][c]){
          continue;
        }else{
          break;
        }
      }
    }
    /**************************FALL THREW**************************************/









    /*************************************************************************/
  }
}
//The least error prone.

//Este programa é aplicado para botoneiras start/stop,
//aplicado em minha casa. Vamos ver duração
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#include<avr/interrupt.h>
#define TRUE 1
#define FALSE 0
#define column 4
```

```
#define lines 52

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B11111111;
  DDRB = B11111111;
  PORTB = B00000000;
}

void loop()
{
  int i, l, c;
  int data[5];
  int Hist[5];
  int treat;
  int count=0;
  //inic key
  data[0] = 0;
  data[1] = 63;
  data[2] = 0;
  data[4] = 0;
  //mem prepared for depth 2 in FSM (finite state machine).
  const int mem[(lines+1)][(column+1)]=
  {
    { 0, 0,65,300, 0},//timer off output 0
    { 0, 0, 0,21, 0},//1
    { 0, 0,65,21,65},//2
    { 0, 0, 0,17,65},//3timer on and output 1
    { 0, 0,65,17,65},//4
    { 0, 0, 0,20,65},//5
    { 0, 0,65,20,65},//6
    { 0, 0, 0, 5,65},//7
    { 0, 0,65, 5,65},//8
    { 0, 0,65,29, 0},//9
    { 0, 0, 0,29, 0},//10
    { 0, 0,65,23, 0},//11
    { 0, 0, 0,23, 0},//12
    { 0, 0,65,31, 0},//13
    { 0, 0, 0,31, 0},//14
    { 0, 0,65, 7, 0},//15
    { 0, 0, 0, 7, 0},//16
    { 0, 0,65,13, 0},//17
    { 0, 0, 0,13, 0},//18
    { 0, 0,65,37, 0},//19
    { 0, 0, 0,37, 0},//20
    { 0, 0,65,28, 0},//21
```

```
  { 0, 0, 0,28, 0},//22
  { 0, 0,65,25, 0},//23
  { 0, 0, 0,25, 0},//24
  { 0, 0,65,17,65},//25
  { 0, 0, 0,13, 0},//26
  { 0, 0, 0,53, 0},//27
  { 0, 0,65,53, 0},//28
  { 0, 0, 0,53, 0},//29
  { 0, 0,65,22, 0},//30
  { 0, 0, 0,22, 0},//31
  { 0, 0,65,61, 0},//32
  { 0, 0, 0,61, 0},//33
  { 0, 0,65,55, 0},//34
  { 0, 0, 0,55, 0},//35
  { 0, 0,65,63, 0},//36
  { 0, 0, 0,63, 0},//37
  { 0, 0,65,52, 0},//38
  { 0, 0, 0,52, 0},//39
  { 0, 0,65,19, 0},//40
  { 0, 0, 0,19, 0},//41
  { 0, 0,65, 9, 0},//42
  { 0, 0, 0, 9, 0},//43
  { 0, 0,65,12, 0},//44
  { 0, 0, 0,12, 0},//45
  { 0, 0, 0,24, 0},//46
  { 0, 0, 0,24, 0},//47
  { 0, 0,65,49, 0},//48
  { 0, 0, 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[1] = data[1], Hist[2] = data[2], count = 0; TRUE; Hist[3] = data[3]){

  treat = 63 & PINC;
  data[3] = treat;

  delay(60);

  //if(!(Serial.available()>0)){
    //Serial.print(data[3]);
    //Serial.print(" , ");
    //Serial.println(count);
  //}

  //timer setup
  if(data[4] == 65){
    count++;
  }else{
    count = 0;
```

```
    }
    //catch timer
    if(count == 15000){
      data[3] = 300;
      count = 0;
    }
    //end timer

    /**********************/
    if(data[3] == Hist[3])
      continue;
    //if(!(Serial.available()>0)){
      //Serial.println(data[3]);
    //}
//    if(data[1] == data[3])
//      continue;
    /**********************/
    /**********************Search and apply changes**********************/
    for( l = 0, c = 0; l < lines   ; l++){
      if(c >= column){
        l--;
        data[4] = mem[l][c];
        //update
        data[0] = data[2];
        data[1] = data[3];
        data[2] = data[4];
        Hist[0] = data[0];
        Hist[1] = data[1];
        Hist[2] = data[2];
        Hist[4] = data[4];
        //send
        PORTB = 63 & data[4];
        break;
      }
      //startup c=2, c=0, for more precise.
      for(c=2; c < column; c++){
        if(data[c] == mem[l][c]){
          continue;
        }else{
          break;
        }
      }
    }
    /**************************FALL THREW**************************************/
```

```
   /*************************************************************************/
  }
}
//Nao existe futuro apenas present proximo.
//The more in depth the state machine is the more obidiente and dumb it becomes, because
//it will have to specify all possible cases.

//Este programa é aplicado para botoneiras start/stop,
//eplicado em minha casa. Vamos ver duração
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#define TRUE 1
#define FALSE 0
#define column 4
#define lines 52

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B11111111;
  DDRB = B11111111;
  PORTB = B00000000;
}

void loop()
{
  int i, l, c;
  int data[5];
  int Hist[5];
  int count;
  //inic key
  data[0] = 0;
  data[1] = 63;
  data[2] = 0;
  data[4] = 0;
  //mem prepared for depth 2 in FSM (finite state machine).
  const int mem[(lines+1)][(column+1)]=
  {
    { 0, 0, 1,300, 0},//timer 23
    { 0, 0, 0,21, 0},//1
    { 0, 0, 1,21, 1},//2
    { 0, 0, 0,17, 1},//3
    { 0, 0, 1,17, 1},//4
    { 0, 0, 0,20, 1},//5
    { 0, 0, 1,20, 1},//6
```

```
  { 0, 0, 0, 5, 1},//7
  { 0, 0, 1, 5, 1},//8
  { 0, 0, 1,29, 0},//9
  { 0, 0, 0,29, 0},//10
  { 0, 0, 1,23, 0},//11
  { 0, 0, 0,23, 0},//12
  { 0, 0, 1,31, 0},//13
  { 0, 0, 0,31, 0},//14
  { 0, 0, 1, 7, 0},//15
  { 0, 0, 0, 7, 0},//16
  { 0, 0, 1,13, 0},//17
  { 0, 0, 0,13, 0},//18
  { 0, 0, 1,37, 0},//19
  { 0, 0, 0,37, 0},//20
  { 0, 0, 1,28, 0},//21
  { 0, 0, 0,28, 0},//22
  { 0, 0, 1,25, 0},//23
  { 0, 0, 0,25, 0},//24
  { 0, 0, 1,17, 1},//25
  { 0, 0, 0,13, 0},//26
  { 0, 0, 0,53, 0},//27
  { 0, 0, 1,53, 0},//28
  { 0, 0, 0,53, 0},//29
  { 0, 0, 1,22, 0},//30
  { 0, 0, 0,22, 0},//31
  { 0, 0, 1,61, 0},//32
  { 0, 0, 0,61, 0},//33
  { 0, 0, 1,55, 0},//34
  { 0, 0, 0,55, 0},//35
  { 0, 0, 1,63, 0},//36
  { 0, 0, 0,63, 0},//37
  { 0, 0, 1,52, 0},//38
  { 0, 0, 0,52, 0},//39
  { 0, 0, 1,19, 0},//40
  { 0, 0, 0,19, 0},//41
  { 0, 0, 1, 9, 0},//42
  { 0, 0, 0, 9, 0},//43
  { 0, 0, 1,12, 0},//44
  { 0, 0, 0,12, 0},//45
  { 0, 0, 0,24, 0},//46
  { 0, 0, 0,24, 0},//47
  { 0, 0, 1,49, 0},//48
  { 0, 0, 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[1] = data[1], Hist[2] = data[2], count = 0; TRUE; Hist[3] = data[3]){

  data[3] = PINC;
```

```
delay(60);

//if(!(Serial.available()>0)){
  //Serial.print(data[3]);
  //Serial.print(" , ");
  //Serial.println(count);
//}

//timer setup
if(data[4] == 1){
  count++;
}else{
  count = 0;
}
//catch timer
if(count == 15000){
  data[3] = 300;
  count = 0;
}
//end timer

/**********************/
if(data[3] == Hist[3])
  continue;
if(data[1] == data[3])
  continue;
/**********************/
/**********************Search and apply changes**********************/
for( l = 0, c = 0; l < lines   ; l++){
  if(c >= column){
    l--;
    data[4] = mem[l][c];
    PORTB = data[4];
    //update
    data[0] = data[2];
    data[1] = data[3];
    data[2] = data[4];
    Hist[0] = data[0];
    Hist[1] = data[1];
    Hist[2] = data[2];
    Hist[4] = data[4];
    break;
  }
  //startup c=2, c=0, for more precise.
  for(c=2; c < column; c++){
    if(data[c] == mem[l][c]){
      continue;
    }else{
```

```
        break;
       }
      }
     }
    /*************************FALL THREW*************************************/









    /************************************************************************/
  }
}
//Nao existe futuro apenas present proximo.
//The more in depth the state machine is the more obidiente and dumb it becomes, because
//it will have to specify all possible cases.

//Este programa é aplicado para botoneiras start/stop,
//aplicado em minha casa. Vamos ver duração
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#define TRUE 1
#define FALSE 0
#define column 4
#define lines 52

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B11111111;
  DDRB = B11111111;
  PORTB = B00000000;
}

void loop()
{
  int i, l, c;
  int data[5];
  int Hist[5];
  int count;
  //inic key
  data[0] = 0;
  data[1] = 63;
  data[2] = 0;
  data[4] = 0;
```

```
//mem prepared for depth 2 in FSM (finite state machine).
const int mem[(lines+1)][(column+1)]=
{
  { 0, 0,65,300, 0},//timer 23
  { 0, 0, 0,21, 0},//1
  { 0, 0,65,21,65},//2
  { 0, 0, 0,17,65},//3
  { 0, 0,65,17,65},//4
  { 0, 0, 0,20,65},//5
  { 0, 0,65,20,65},//6
  { 0, 0, 0, 5,65},//7
  { 0, 0,65, 5,65},//8
  { 0, 0,65,29, 0},//9
  { 0, 0, 0,29, 0},//10
  { 0, 0,65,23, 0},//11
  { 0, 0, 0,23, 0},//12
  { 0, 0,65,31, 0},//13
  { 0, 0, 0,31, 0},//14
  { 0, 0,65, 7, 0},//15
  { 0, 0, 0, 7, 0},//16
  { 0, 0,65,13, 0},//17
  { 0, 0, 0,13, 0},//18
  { 0, 0,65,37, 0},//19
  { 0, 0, 0,37, 0},//20
  { 0, 0,65,28, 0},//21
  { 0, 0, 0,28, 0},//22
  { 0, 0,65,25, 0},//23
  { 0, 0, 0,25, 0},//24
  { 0, 0,65,17,65},//25
  { 0, 0, 0,13, 0},//26
  { 0, 0, 0,53, 0},//27
  { 0, 0,65,53, 0},//28
  { 0, 0, 0,53, 0},//29
  { 0, 0,65,22, 0},//30
  { 0, 0, 0,22, 0},//31
  { 0, 0,65,61, 0},//32
  { 0, 0, 0,61, 0},//33
  { 0, 0,65,55, 0},//34
  { 0, 0, 0,55, 0},//35
  { 0, 0,65,63, 0},//36
  { 0, 0, 0,63, 0},//37
  { 0, 0,65,52, 0},//38
  { 0, 0, 0,52, 0},//39
  { 0, 0,65,19, 0},//40
  { 0, 0, 0,19, 0},//41
  { 0, 0,65, 9, 0},//42
  { 0, 0, 0, 9, 0},//43
  { 0, 0,65,12, 0},//44
  { 0, 0, 0,12, 0},//45
```

```
  { 0, 0, 0,24, 0},//46
  { 0, 0, 0,24, 0},//47
  { 0, 0,65,49, 0},//48
  { 0, 0, 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[1] = data[1], Hist[2] = data[2], count = 0; TRUE; Hist[3] = data[3]){

  data[3] = PINC;

  delay(60);

  //if(!(Serial.available()>0)){
    //Serial.print(data[3]);
    //Serial.print(" , ");
    //Serial.println(count);
  //}

  //timer setup
  if(data[4] == 65){
    count++;
  }else{
    count = 0;
  }
  //catch timer
  if(count == 15000){
    PORTB = 0;
    count = 0;
  }
  //end timer

  /**********************/
  if(data[3] == Hist[3])
    continue;
  if(data[1] == data[3])
    continue;
  /**********************/
  /**********************Search and apply changes**********************/
  for( l = 0, c = 0; l < lines   ; l++){
    if(c >= column){
      l--;
      data[4] = mem[l][c];
      //update
      data[0] = data[2];
      data[1] = data[3];
      data[2] = data[4];
      Hist[0] = data[0];
      Hist[1] = data[1];
```

```
      Hist[2] = data[2];
      Hist[4] = data[4];
      //send
      PORTB = 63 & data[4];
      break;
    }
    //startup c=2, c=0, for more precise.
    for(c=2; c < column; c++){
      if(data[c] == mem[l][c]){
        continue;
      }else{
        break;
      }
    }
  }
  /**************************FALL THREW**************************************/




  /*****************************************************************************/
 }
}
//Nao existe futuro apenas present proximo.
//The more in depth the state machine is the more obidiente and dumb it becomes, because
//it will have to specify all possible cases.

//Este programa é aplicado para botoneiras start/stop,
//aplicado em minha casa. Vamos ver duração
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#include<avr/interrupt.h>
#define TRUE 1
#define FALSE 0
#define column 4
#define lines 52

void setup()
{
 // start serial port at 9600 bps:
 //Serial.begin(9600);
 DDRC = B00000000;
 PORTC = B11111111;
 DDRB = B11111111;
 PORTB = B00000000;
}
```

```
void loop()
{
  int i, l, c;
  int data[5];
  int Hist[5];
  int treat;
  int count=0;
  //inic key
  data[0] = 0;
  data[1] = 63;
  data[2] = 0;
  data[4] = 0;
  //mem prepared for depth 2 in FSM (finite state machine).
  const int mem[(lines+1)][(column+1)]=
  {
    { 0, 0,65,300, 0},//timer off output 0
    { 0, 0, 0,21, 0},//1
    { 0, 0,65,21,65},//2
    { 0, 0, 0,17,65},//3timer on and output 1
    { 0, 0,65,17,65},//4
    { 0, 0, 0,20,65},//5
    { 0, 0,65,20,65},//6
    { 0, 0, 0, 5,65},//7
    { 0, 0,65, 5,65},//8
    { 0, 0,65,29, 0},//9
    { 0, 0, 0,29, 0},//10
    { 0, 0,65,23, 0},//11
    { 0, 0, 0,23, 0},//12
    { 0, 0,65,31, 0},//13
    { 0, 0, 0,31, 0},//14
    { 0, 0,65, 7, 0},//15
    { 0, 0, 0, 7, 0},//16
    { 0, 0,65,13, 0},//17
    { 0, 0, 0,13, 0},//18
    { 0, 0,65,37, 0},//19
    { 0, 0, 0,37, 0},//20
    { 0, 0,65,28, 0},//21
    { 0, 0, 0,28, 0},//22
    { 0, 0,65,25, 0},//23
    { 0, 0, 0,25, 0},//24
    { 0, 0,65,17,65},//25
    { 0, 0, 0,13, 0},//26
    { 0, 0, 0,53, 0},//27
    { 0, 0,65,53, 0},//28
    { 0, 0, 0,53, 0},//29
    { 0, 0,65,22, 0},//30
    { 0, 0, 0,22, 0},//31
    { 0, 0,65,61, 0},//32
```

```
  { 0, 0, 0,61, 0},//33
  { 0, 0,65,55, 0},//34
  { 0, 0, 0,55, 0},//35
  { 0, 0,65,63, 0},//36
  { 0, 0, 0,63, 0},//37
  { 0, 0,65,52, 0},//38
  { 0, 0, 0,52, 0},//39
  { 0, 0,65,19, 0},//40
  { 0, 0, 0,19, 0},//41
  { 0, 0,65, 9, 0},//42
  { 0, 0, 0, 9, 0},//43
  { 0, 0,65,12, 0},//44
  { 0, 0, 0,12, 0},//45
  { 0, 0, 0,24, 0},//46
  { 0, 0, 0,24, 0},//47
  { 0, 0,65,49, 0},//48
  { 0, 0, 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[1] = data[1], Hist[2] = data[2], count = 0; TRUE; Hist[3] = data[3]){

  treat = 63 & PINC;
  data[3] = treat;

  delay(60);

  //if(!(Serial.available()>0)){
    //Serial.print(data[3]);
    //Serial.print(" , ");
    //Serial.println(count);
  //}

  //timer setup
  if(data[4] == 65){
    count++;
  }else{
    count = 0;
  }
  //catch timer
  if(count == 15000){
    data[3] = 300;
    count = 0;
  }
  //end timer

  /*********************/
  if(data[3] == Hist[3])
    continue;
```

```
    //if(!(Serial.available()>0)){
      //Serial.println(data[3]);
    //}
//    if(data[1] == data[3])
//      continue;
    /*********************/
    /**********************Search and apply changes*********************/
    for( l = 0, c = 0; l < lines   ; l++){
      if(c >= column){
        l--;
        data[4] = mem[l][c];
        //update
        data[0] = data[2];
        data[1] = data[3];
        data[2] = data[4];
        Hist[0] = data[0];
        Hist[1] = data[1];
        Hist[2] = data[2];
        Hist[4] = data[4];
        //send
        PORTB = 63 & data[4];
        break;
      }
      //startup c=2, c=0, for more precise.
      for(c=2; c < column; c++){
        if(data[c] == mem[l][c]){
          continue;
        }else{
          break;
        }
      }
    }
    /**************************FALL THREW*************************************/









    /*************************************************************************/
  }
}
//Nao existe futuro apenas present proximo.
//The more in depth the state machine is the more obidiente and dumb it becomes, because
//it will have to specify all possible cases.

//Este programa é aplicado para botoneiras start/stop,
//aplicado em minha casa. Vamos ver duração
```

```
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#include<avr/interrupt.h>
#define TRUE 1
#define FALSE 0
#define column 4
#define lines 52

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B11111111;
  DDRB = B11111111;
  PORTB = B00000000;
}

void loop()
{
  int i, l, c;
  int data[5];
  int Hist[5];
  int count=0;
  //inic key
  data[0] = 0;
  data[1] = 63;
  data[2] = 0;
  data[4] = 0;
  //mem prepared for depth 2 in FSM (finite state machine).
  const int mem[(lines+1)][(column+1)]=
  {
    { 0, 0,65,300, 0},//timer off
    { 0, 0, 0,21, 0},//1
    { 0, 0,65,21,65},//2
    { 0, 0, 0,17,65},//3timer on and output 1
    { 0, 0,65,17,65},//4
    { 0, 0, 0,20,65},//5
    { 0, 0,65,20,65},//6
    { 0, 0, 0, 5,65},//7
    { 0, 0,65, 5,65},//8
    { 0, 0,65,29, 0},//9
    { 0, 0, 0,29, 0},//10
    { 0, 0,65,23, 0},//11
    { 0, 0, 0,23, 0},//12
    { 0, 0,65,31, 0},//13
    { 0, 0, 0,31, 0},//14
    { 0, 0,65, 7, 0},//15
    { 0, 0, 0, 7, 0},//16
```

```
  { 0, 0,65,13, 0},//17
  { 0, 0, 0,13, 0},//18
  { 0, 0,65,37, 0},//19
  { 0, 0, 0,37, 0},//20
  { 0, 0,65,28, 0},//21
  { 0, 0, 0,28, 0},//22
  { 0, 0,65,25, 0},//23
  { 0, 0, 0,25, 0},//24
  { 0, 0,65,17,65},//25
  { 0, 0, 0,13, 0},//26
  { 0, 0, 0,53, 0},//27
  { 0, 0,65,53, 0},//28
  { 0, 0, 0,53, 0},//29
  { 0, 0,65,22, 0},//30
  { 0, 0, 0,22, 0},//31
  { 0, 0,65,61, 0},//32
  { 0, 0, 0,61, 0},//33
  { 0, 0,65,55, 0},//34
  { 0, 0, 0,55, 0},//35
  { 0, 0,65,63, 0},//36
  { 0, 0, 0,63, 0},//37
  { 0, 0,65,52, 0},//38
  { 0, 0, 0,52, 0},//39
  { 0, 0,65,19, 0},//40
  { 0, 0, 0,19, 0},//41
  { 0, 0,65, 9, 0},//42
  { 0, 0, 0, 9, 0},//43
  { 0, 0,65,12, 0},//44
  { 0, 0, 0,12, 0},//45
  { 0, 0, 0,24, 0},//46
  { 0, 0, 0,24, 0},//47
  { 0, 0,65,49, 0},//48
  { 0, 0, 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[1] = data[1], Hist[2] = data[2], count = 0; TRUE; Hist[3] = data[3]){

  data[3] = 63 & PINC;

  delay(60);

  //if(!(Serial.available()>0)){
    //Serial.print(data[3]);
    //Serial.print(" , ");
    //Serial.println(count);
  //}

  //timer setup
```

```
if(data[4] == 65){
  count++;
}else{
  count = 0;
}
//catch timer
if(count == 15000){
  data[3] = 300;
  count = 0;
}
//end timer

/**********************/
if(data[3] == Hist[3])
  continue;
if(data[1] == data[3])
  continue;
/**********************/
/**********************Search and apply changes*********************/
for( l = 0, c = 0; l < lines   ; l++){
  if(c >= column){
    l--;
    data[4] = mem[l][c];
    //update
    data[0] = data[2];
    data[1] = data[3];
    data[2] = data[4];
    Hist[0] = data[0];
    Hist[1] = data[1];
    Hist[2] = data[2];
    Hist[4] = data[4];
    //send
    PORTB = 63 & data[4];
    break;
  }
  //startup c=2, c=0, for more precise.
  for(c=2; c < column; c++){
    if(data[c] == mem[l][c]){
      continue;
    }else{
      break;
    }
  }
}
/*************************FALL THREW*************************************/
```

```
   /*************************************************************************/
  }
}
//Nao existe futuro apenas present proximo.
//The more in depth the state machine is the more obidiente and dumb it becomes, because
//it will have to specify all possible cases.

//Este programa é aplicado para botoneiras start/stop,
//aplicado em minha casa. Vamos ver duração
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#define TRUE 1
#define FALSE 0
#define column 4
#define lines 52

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B11111111;
  DDRB = B11111111;
  PORTB = B00000000;
}

void loop()
{
  int i, l, c;
  int data[5];
  int Hist[5];
  int count;
  //inic key
  data[0] = 0;
  data[1] = 63;
  data[2] = 0;
  data[4] = 0;
  //mem prepared for depth 2 in FSM (finite state machine).
  const int mem[(lines+1)][(column+1)]=
  {
    { 0, 0, 1,300, 0},//timer 23
    { 0, 0, 0,21, 0},//1
    { 0, 0, 1,21, 1},//2
    { 0, 0, 0,17, 1},//3
    { 0, 0, 1,17, 1},//4
    { 0, 0, 0,20, 1},//5
```

```
  { 0, 0, 1,20, 1},//6
  { 0, 0, 0, 5, 1},//7
  { 0, 0, 1, 5, 1},//8
  { 0, 0, 1,29, 0},//9
  { 0, 0, 0,29, 0},//10
  { 0, 0, 1,23, 0},//11
  { 0, 0, 0,23, 0},//12
  { 0, 0, 1,31, 0},//13
  { 0, 0, 0,31, 0},//14
  { 0, 0, 1, 7, 0},//15
  { 0, 0, 0, 7, 0},//16
  { 0, 0, 1,13, 0},//17
  { 0, 0, 0,13, 0},//18
  { 0, 0, 1,37, 0},//19
  { 0, 0, 0,37, 0},//20
  { 0, 0, 1,28, 0},//21
  { 0, 0, 0,28, 0},//22
  { 0, 0, 1,25, 0},//23
  { 0, 0, 0,25, 0},//24
  { 0, 0, 1,17, 1},//25
  { 0, 0, 0,13, 0},//26
  { 0, 0, 0,53, 0},//27
  { 0, 0, 1,53, 0},//28
  { 0, 0, 0,53, 0},//29
  { 0, 0, 1,22, 0},//30
  { 0, 0, 0,22, 0},//31
  { 0, 0, 1,61, 0},//32
  { 0, 0, 0,61, 0},//33
  { 0, 0, 1,55, 0},//34
  { 0, 0, 0,55, 0},//35
  { 0, 0, 1,63, 0},//36
  { 0, 0, 0,63, 0},//37
  { 0, 0, 1,52, 0},//38
  { 0, 0, 0,52, 0},//39
  { 0, 0, 1,19, 0},//40
  { 0, 0, 0,19, 0},//41
  { 0, 0, 1, 9, 0},//42
  { 0, 0, 0, 9, 0},//43
  { 0, 0, 1,12, 0},//44
  { 0, 0, 0,12, 0},//45
  { 0, 0, 0,24, 0},//46
  { 0, 0, 0,24, 0},//47
  { 0, 0, 1,49, 0},//48
  { 0, 0, 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[1] = data[1], Hist[2] = data[2], count = 0; TRUE; Hist[3] = data[3]){
```

```
data[3] = PINC;

delay(60);

//if(!(Serial.available()>0)){
  //Serial.print(data[3]);
  //Serial.print(" , ");
  //Serial.println(count);
//}

//timer setup
if(data[4] == 1){
  count++;
}else{
  count = 0;
}
//catch timer
if(count == 15000){
  data[3] = 300;
  count = 0;
}
//end timer

/**********************/
if(data[3] == Hist[3])
  continue;
if(data[1] == data[3])
  continue;
/**********************/
/**********************Search and apply changes**********************/
for( l = 0, c = 0; l < lines   ; l++){
  if(c >= column){
    l--;
    data[4] = mem[l][c];
    //update
    data[0] = data[2];
    data[1] = data[3];
    data[2] = data[4];
    Hist[0] = data[0];
    Hist[1] = data[1];
    Hist[2] = data[2];
    Hist[4] = data[4];
    //send
    PORTB = 63 & data[4];
    break;
  }
  //startup c=2, c=0, for more precise.
  for(c=2; c < column; c++){
    if(data[c] == mem[l][c]){
```

```
        continue;
       }else{
        break;
       }
      }
    }
    /*************************FALL THREW*************************************/









    /************************************************************************/
  }
}
//Nao existe futuro apenas present proximo.
//The more in depth the state machine is the more obidiente and dumb it becomes, because
//it will have to specify all possible cases.

//Este programa é aplicado para botoneiras start/stop,
//eplicado em minha casa. Vamos ver duração
//teste endurance de placa arduino duemialnove.
#include<avr/io.h>
#define TRUE 1
#define FALSE 0
#define column 4
#define lines 52

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B11111111;
  DDRB = B11111111;
  PORTB = B00000000;
}

void loop()
{
  int i, l, c;
  int data[5];
  int Hist[5];
  int count;
  //inic key
  data[0] = 0;
  data[1] = 63;
```

```
data[2] = 0;
data[4] = 0;
//mem prepared for depth 2 in FSM (finite state machine).
const int mem[(lines+1)][(column+1)]=
{
  { 0, 0, 1,300, 0},//timer 23
  { 0, 0, 0,21, 0},//1
  { 0, 0, 1,21, 1},//2
  { 0, 0, 0,17, 1},//3
  { 0, 0, 1,17, 1},//4
  { 0, 0, 0,20, 1},//5
  { 0, 0, 1,20, 1},//6
  { 0, 0, 0, 5, 1},//7
  { 0, 0, 1, 5, 1},//8
  { 0, 0, 1,29, 0},//9
  { 0, 0, 0,29, 0},//10
  { 0, 0, 1,23, 0},//11
  { 0, 0, 0,23, 0},//12
  { 0, 0, 1,31, 0},//13
  { 0, 0, 0,31, 0},//14
  { 0, 0, 1, 7, 0},//15
  { 0, 0, 0, 7, 0},//16
  { 0, 0, 1,13, 0},//17
  { 0, 0, 0,13, 0},//18
  { 0, 0, 1,37, 0},//19
  { 0, 0, 0,37, 0},//20
  { 0, 0, 1,28, 0},//21
  { 0, 0, 0,28, 0},//22
  { 0, 0, 1,25, 0},//23
  { 0, 0, 0,25, 0},//24
  { 0, 0, 1,17, 1},//25
  { 0, 0, 0,13, 0},//26
  { 0, 0, 0,53, 0},//27
  { 0, 0, 1,53, 0},//28
  { 0, 0, 0,53, 0},//29
  { 0, 0, 1,22, 0},//30
  { 0, 0, 0,22, 0},//31
  { 0, 0, 1,61, 0},//32
  { 0, 0, 0,61, 0},//33
  { 0, 0, 1,55, 0},//34
  { 0, 0, 0,55, 0},//35
  { 0, 0, 1,63, 0},//36
  { 0, 0, 0,63, 0},//37
  { 0, 0, 1,52, 0},//38
  { 0, 0, 0,52, 0},//39
  { 0, 0, 1,19, 0},//40
  { 0, 0, 0,19, 0},//41
  { 0, 0, 1, 9, 0},//42
  { 0, 0, 0, 9, 0},//43
```

```
  { 0, 0, 1,12, 0},//44
  { 0, 0, 0,12, 0},//45
  { 0, 0, 0,24, 0},//46
  { 0, 0, 0,24, 0},//47
  { 0, 0, 1,49, 0},//48
  { 0, 0, 0,49, 0}//49
};

/*******************CICLOS DE MAQUINA*********************/
for( Hist[0] = data[0], Hist[1] = data[1], Hist[2] = data[2], count = 0; TRUE; Hist[3] = data[3]){

  data[3] = PINC;

  delay(60);

  //if(!(Serial.available()>0)){
    //Serial.print(data[3]);
    //Serial.print(" , ");
    //Serial.println(count);
  //}

  //timer setup
  if(data[4] == 1){
    count++;
  }else{
    count = 0;
  }
  //catch timer
  if(count == 15000){
    data[3] = 300;
    count = 0;
  }
  //end timer

  /**********************/
  if(data[3] == Hist[3])
    continue;
  if(data[1] == data[3])
    continue;
  /**********************/
  /**********************Search and apply changes**********************/
  for( l = 0, c = 0; l < lines   ; l++){
    if(c >= column){
      l--;
      data[4] = mem[l][c];
      PORTB = data[4];
      //update
      data[0] = data[2];
      data[1] = data[3];
```

```
        data[2] = data[4];
        Hist[0] = data[0];
        Hist[1] = data[1];
        Hist[2] = data[2];
        Hist[4] = data[4];
        break;
      }
      //startup c=2, c=0, for more precise.
      for(c=2; c < column; c++){
        if(data[c] == mem[l][c]){
          continue;
        }else{
          break;
        }
      }
    }
    /*************************FALL THREW*************************************/









    /************************************************************************/
  }
}
//Nao existe futuro apenas present proximo.
//The more in depth the state machine is the more obidiente and dumb it becomes, because
//it will have to specify all possible cases.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
```

```
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  for(;TRUE;Hist[B]=Entry[B]){
    //Hist[C]=Entry[C],
    //Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    //if(Entry[C] != Hist[C])
      //Serial.println(Entry[C],DEC);
    //delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
```

```
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char* x);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B11111111;
  PORTB=B00000000;

}
//begin

void loop()
{
  uint8_t i=0;
```

```
  uint8_t Entry[2];
  uint8_t Hist[2];
  char IncomingByte;
  char State[BufSize];
  //INIC
  for( Hist[C]=0,Hist[B]=0; TRUE; Hist[C] = Entry[C], Hist[B] = Entry[B]){
    //ENTRADA Pinos
    Entry[C]=PINC;
    //ENTRADA Serial
    if(Serial.available()){
      delay(25);//wait for incoming data.
      for(i = 0; IncomingByte = Serial.read(); i++){
           if((IncomingByte == '\r') || (IncomingByte == '\n')){
             State[i] = '\0';
             Serial.flush();
             break;

           }else{
             State[i]=IncomingByte;

           }

      }
      Entry[B] = getnum(&State[0]);

    }
    if((Entry[C] == Hist[C]) && (Entry[B] == Hist[B]))
      continue;
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    PORTB = Entry[B];
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
```

```
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define lines 57
#define buf_size 32

/***MOME MEMORY***/
const char *mem[lines][2]={
  {"temporizador","0"},
  {"0:0","65"},
  {"0:16","65"},
  {"0:1","65"},
  {"0:4","65"},
  {"0:17","65"},
  {"0:20","65"},
  {"0:5","65"},
  {"65:28","0"},
  {"65:29","0"},
  {"65:23","0"},
  {"65:31","0"},
  {"65:7","0"},
  {"65:12","0"},
  {"65:13","0"},
  {"65:36","0"},
  {"65:32","0"},
  {"65:33","0"},
  {"65:37","0"},
  {"65:39","0"},
  {"65:40","0"},
  {"65:41","0"},
```

```
  {"65:28","0"},
  {"65:24","0"},
  {"65:25","0"},
  {"65:56","0"},
  {"65:57","0"},
  {"65:56","0"},
  {"65:2","0"},
  {"65:34","0"},
  {"65:35","0"},
  {"65:3","0"},
  {"65:25","0"},
  {"65:53","0"},
  {"65:22","0"},
  {"65:58","0"},
  {"65:59","0"},
  {"65:60","0"},
  {"65:61","0"},
  {"65:54","0"},
  {"65:55","0"},
  {"65:62","0"},
  {"65:63","0"},
  {"65:42","0"},
  {"65:43","0"},
  {"65:44","0"},
  {"65:45","0"},
  {"65:46","0"},
  {"65:47","0"},
  {"65:50","0"},
  {"65:51","0"},
  {"65:52","0"},
  {"65:18","0"},
  {"65:19","0"},
  {"65:9","0"},
  {"65:12","0"},
  {"65:49","0"}
};


//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);
char *LMOME(char *memory[lines][2],char keygen[2][buf_size],char input[buf_size]);
int intostr(int value,char *x);
void setup()
{
  //start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
```

```
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  char keygen[2][buf_size];
  char hist[buf_size];
  char input[buf_size];
  char Input[buf_size];
  char response[buf_size];
  int counter;
  //inic key
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"0");
  //mem prepared for depth 2 in FSM (finite state machine).

  /*******************CICLOS DE MAQUINA**********************/
  for(counter=0;TRUE;strcpy(hist,Input)){

    //strcpy(input,63 & PINC);
    intostr(63 & PINC,input);
    delay(60);

    //logical key
    strcpy(Input,keygen[0]);
    strcat(Input,":");
    strcat(Input,input);

    //if(!(Serial.available()>0)){
      //Serial.print(Input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(!strcmp(keygen[0],"65")){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter==150){
      strcpy(Input,"temporizador");
      counter = 0;
    }//15000
    //end timer
```

```
    /*********************/
    if(!strcmp(Input,hist))
      continue;
    //if(!(Serial.available()>0)){
      //Serial.println(Input);
    //}

    /***************Search and apply changes***************/

    strcpy(response,LMOME(mem,keygen,Input));

    PORTB=63 & atoi(response);

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/
```

```
//LMOME from matrix
char *LMOME(const char *memory[lines][2],char keygen[2][buf_size],char input[buf_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/***/
int intostr(int value,char *x)
{
  int i;
  int temp=value;
  for(i=0;temp!=0;temp/=10,i++);
  x[i]='\0';
  for(i--,temp=value,temp%=10;!(i<0);x[i]=temp+48,value/=10,temp=value,i--,temp%=10);
  return 0;
}

//& bitwise AND, && logical bool and
//solving problems using complicated
//methods make it more viable and strong.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
```

```
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(30);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(30);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }
```

```
}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lmoore 10
#define lmealy 11
#define TRUE 1

/***Mealy Memory***/
int memmealy[lmealy][4]={
  {0,0,63,0},
  {0,0,61,1},
  {0,0,62,32},
  {63,0,62,32},
  {62,32,63,32},
  {63,32,62,0},
  {62,0,63,0},
  {63,0,61,33},
  {61,33,63,33},
  {63,33,62,0},
  {63,32,60,0}
};
```

```
/***Moore Memory***/
int memmoore[lmoore][3]={
  {0,63,0},
  {0,62,32},
  {0,61,1},
  {32,63,32},
  {32,62,0},
  {0,63,0},
  {0,61,33},
  {33,63,33},
  {33,62,0},
  {32,60,0}
};

int MOORE(int mem[lmoore][3],int keygen[2],int input);
int MEALY(int mem[lmealy][4],int keygen[2],int input);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;
  keygen[1]=0;

  /*****************************************/

  for(hist=0;TRUE;hist=input){

    //for logic
    //keygen[0]=0;
    //keygen[1]=0;
    //PORTS
    input=PINC;
    delay(30);
```

```
    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/
    response=MEALY(memmealy,keygen,input);

    //response=MOORE(memmoore,keygen,input);
    PORTB=response&B00111111;//masked



    /*********************************************************/
  }//for TRUE
}//loop
/******/
int MOORE(int mem[lmoore][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[0]==input)//evitar redundancia
    return keygen[1];
  for(iterator=0;iterator<lmoore;iterator++){
    keyfound=(mem[iterator][0]==keygen[1] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=mem[iterator][2];
      //communicate
      break;
    }
  }//for iterator
  return keygen[1];
}
/******/
int MEALY(int mem[lmealy][4],int keygen[2],int input)
{
  int iterator;
  int keyfound;//evita redundancia
  for(iterator=0;iterator<lmealy;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==keygen[1] && mem[iterator]
[2]==input);//bool
    if(keyfound){
      //MEALY UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=mem[iterator][3];
      break;
    }
```

```
  }//for iterator
  return keygen[1];
}
/******/

#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lines 13
#define TRUE 1

/***MOME Memory***/
int mem[lines][3]={
  {0,63,0},
  {0,59,33},
  {33,63,0},
  {0,62,32},
  {0,61,1},
  {1,62,0},
  {32,63,32},
  {32,62,0},
  {0,63,0},
  {0,61,33},
  {33,63,33},
  {33,62,0},
  {32,60,0}
};

//PROTOTYPES
int MOME(int mem[lines][3],int keygen[2],int input);
int ReadInt(int nmin, int nmax);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;//output
  keygen[1]=0;//input
```

```
  /*****************************************/

  for(hist=0;TRUE;hist=input){

    //for logic
    //keygen[0]=0;//logic
    //PORTS
    input=PINC;
    delay(30);




    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/


    response=MOME(mem,keygen,input);
    PORTB=response&B00111111;//masked



    /*********************************************************/
  }//for TRUE
}//loop
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
```

```
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/***/

#include<avr/io.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char* x);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B11111111;
  PORTB=B00000000;

}
//begin

void loop()
{
  uint8_t i=0;
  uint8_t flag;
  uint8_t Entry[2];
  uint8_t Past[2];
  uint8_t Hist[2];
  uint8_t num;
  char IncomingByte;
  char State[BufSize];
  //INIC
  for( Past[C] = Entry[C], Past[B] = Entry[B]; TRUE; Hist[C] = Entry[C], Hist[B] = Entry[B]){
```

```
   //ENTRADA Portas
     Entry[C]=PINC;
   //ENTRADA Serial
   if(Serial.available()){
     delay(25);//wait for incoming data.
     for(i = 0; IncomingByte = Serial.read(); i++){
          if((IncomingByte == '\r') || (IncomingByte == '\n')){
            State[i] = '\0';
            Serial.flush();
            break;

          }else{
            State[i]=IncomingByte;

          }

     }
     Entry[B] = getnum(&State[0]);

   }
   if((Entry[C] == Hist[C]) && (Entry[B] == Hist[B]))
     continue;
   for(delay(10) ; TRUE; Past[C] = Entry[C], Past[B] = Entry[B]){
     if((Entry[C] == Past[C]) && (Entry[B] == Past[B]))
          break;
     /**Processing***/
     if(Entry[C] != Past[C]){

        num=Entry[C];
           Serial.println(num,DEC);
        //Serial.write(&Entry[C],1);

     }
     if(Entry[B] != Past[B]){
           PORTB = Entry[B];

     }

   }

  }

}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
```

```
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}

//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define TRUE 1
#define FALSE 0
#define lines 57

/***LMOME MEMORY***/
const int mem[lines][2]={
  {1<<12|1<<6|0,0},
  {0<<6|0,1},
  {0<<6|16,1},
  {0<<6|1,1},
  {0<<6|4,1},
  {0<<6|17,1},
  {0<<6|20,1},
  {0<<6|5,1},
  {1<<6|28,0},
  {1<<6|29,0},
  {1<<6|23,0},
  {1<<6|31,0},
```

```
  {1<<6|7,0},
  {1<<6|12,0},
  {1<<6|13,0},
  {1<<6|36,0},
  {1<<6|32,0},
  {1<<6|33,0},
  {1<<6|37,0},
  {1<<6|39,0},
  {1<<6|40,0},
  {1<<6|41,0},
  {1<<6|28,0},
  {1<<6|24,0},
  {1<<6|25,0},
  {1<<6|56,0},
  {1<<6|57,0},
  {1<<6|56,0},
  {1<<6|2,0},
  {1<<6|34,0},
  {1<<6|35,0},
  {1<<6|3,0},
  {1<<6|25,0},
  {1<<6|53,0},
  {1<<6|22,0},
  {1<<6|58,0},
  {1<<6|59,0},
  {1<<6|60,0},
  {1<<6|61,0},
  {1<<6|54,0},
  {1<<6|55,0},
  {1<<6|62,0},
  {1<<6|63,0},
  {1<<6|42,0},
  {1<<6|43,0},
  {1<<6|44,0},
  {1<<6|45,0},
  {1<<6|46,0},
  {1<<6|47,0},
  {1<<6|50,0},
  {1<<6|51,0},
  {1<<6|52,0},
  {1<<6|18,0},
  {1<<6|19,0},
  {1<<6|9,0},
  {1<<6|12,0},
  {1<<6|49,0}
};

//PROTOTIPOS
int ReadInt(int nmin, int nmax);
```

```
int LMOME(int mem[lines][2],int keygen[2],int input);

void setup()
{
  // start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  int counter;
  //inic key
  keygen[0] = 0;//output
  keygen[1] = 0;//input
  //FSM (finite state machine).

  /*******************CICLOS DE MAQUINA**********************/
  for(counter=0,hist=0;TRUE;hist=input){

    input = keygen[0]<<6|(63 & PINC);//FSM key
    delay(60);

    //if(!(Serial.available()>0)){
      //Serial.print(input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(keygen[0] == 1){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter == 15000){
      input = 1<<12|1<<6|0;
      counter = 0;
    }//15000
    //end timer
```

```
    /*********************/
    if(input == hist)
      continue;
    /****************Search and apply changes***************/

    //if(!(Serial.available()>0)){
      //Serial.println(keygen[0]);
    //}

    response=LMOME(mem,keygen,input);

    PORTB=63 & response;

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int LMOME(const int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/******/
```

```
#include<avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#define lines 13
#define TRUE 1

/***MOME Memory***/
int mem[lines][3]={
  {0,63,0},
  {0,59,33},
  {33,63,0},
  {0,62,32},
  {0,61,1},
  {1,62,0},
  {32,63,32},
  {32,62,0},
  {0,63,0},
  {0,61,33},
  {33,63,33},
  {33,62,0},
  {32,60,0}
};

//PROTOTYPES
int MOME(int mem[lines][3],int keygen[2],int input);
int ReadInt(int nmin, int nmax);

void setup() {
  //Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;
}

void loop()
{
  int keygen[2];
  int hist;
  int input;
  int response;
  //Inicialize first states
  keygen[0]=0;//output
  keygen[1]=0;//input

  /*****************************************/

  for(hist=0;TRUE;hist=input){
```

```
    //for logic
    //keygen[0]=0;//logic
    //PORTS
    input=PINC;
    delay(30);









    if(input==hist)//evitar redundancia
      continue;
    //Serial.println(keygen[2],DEC);
    /*******************Find and Conquer*********************/


    response=MOME(mem,keygen,input);
    PORTB=response&B00111111;//masked



    /*********************************************************/
  }//for TRUE
}//loop
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
```

```
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
/***/

/*
Aplicaçao autoria: sergio santos.
email: sergio.salazar.santos@gmail.com
tele: 916919898
Este programa é aplicado para botoneiras start/stop,
*/
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdlib.h>
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#define lines 57
#define buf_size 32

/***MOME MEMORY***/
const char *mem[lines][2]={
  {"temporizador","0"},
  {"0:0","65"},
  {"0:16","65"},
  {"0:1","65"},
  {"0:4","65"},
  {"0:17","65"},
  {"0:20","65"},
  {"0:5","65"},
  {"65:28","0"},
  {"65:29","0"},
  {"65:23","0"},
  {"65:31","0"},
  {"65:7","0"},
  {"65:12","0"},
  {"65:13","0"},
  {"65:36","0"},
  {"65:32","0"},
  {"65:33","0"},
  {"65:37","0"},
  {"65:39","0"},
  {"65:40","0"},
```

```
  {"65:41","0"},
  {"65:28","0"},
  {"65:24","0"},
  {"65:25","0"},
  {"65:56","0"},
  {"65:57","0"},
  {"65:56","0"},
  {"65:2","0"},
  {"65:34","0"},
  {"65:35","0"},
  {"65:3","0"},
  {"65:25","0"},
  {"65:53","0"},
  {"65:22","0"},
  {"65:58","0"},
  {"65:59","0"},
  {"65:60","0"},
  {"65:61","0"},
  {"65:54","0"},
  {"65:55","0"},
  {"65:62","0"},
  {"65:63","0"},
  {"65:42","0"},
  {"65:43","0"},
  {"65:44","0"},
  {"65:45","0"},
  {"65:46","0"},
  {"65:47","0"},
  {"65:50","0"},
  {"65:51","0"},
  {"65:52","0"},
  {"65:18","0"},
  {"65:19","0"},
  {"65:9","0"},
  {"65:12","0"},
  {"65:49","0"}
};


//PROTOTIPOS
int ReadInt(int nmin, int nmax);
int MOME(int mem[lines][3],int keygen[2],int input);
char *LMOME(char *memory[lines][2],char keygen[2][buf_size],char input[buf_size]);
char *intostr(int value);
void setup()
{
  //start serial port at 9600 bps:
  //Serial.begin(9600);
  DDRC = B00000000;
```

```
  PORTC = B00111111;
  DDRB = B00111111;
  PORTB = B00000000;
}

void loop()
{
  char keygen[2][buf_size];
  char hist[buf_size];
  int input;
  char Input[buf_size];
  char response[buf_size];
  int counter;
  //inic key
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"0");
  //mem prepared for depth 2 in FSM (finite state machine).

  /*******************CICLOS DE MAQUINA*********************/
  for(counter=0;TRUE;strcpy(hist,Input)){

    //strcpy(input,63 & PINC);
    input=63 & PINC;
    delay(60);

    //logical key
    strcpy(Input,keygen[0]);
    strcat(Input,":");
    strcat(Input,intostr(input));

    //if(!(Serial.available()>0)){
      //Serial.print(Input);
      //Serial.print(" , ");
      //Serial.println(counter);
    //}

    //timer setup
    if(!strcmp(keygen[0],"65")){
      counter++;
    }else{
      counter = 0;
    }
    //catch timer
    if(counter==150){
      strcpy(Input,"temporizador");
      counter = 0;
    }//15000
    //end timer
```

```
    /**********************/
    if(!strcmp(Input,hist))
      continue;
    //if(!(Serial.available()>0)){
      //Serial.println(Input);
    //}

    /***************Search and apply changes***************/

    strcpy(response,LMOME(mem,keygen,Input));

    PORTB=63 & atoi(response);

    /*********************************************************/
  }
}
/******/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
}
/******/
int MOME(const int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
```

```
/******/
//LMOME from matrix
char *LMOME(const char *memory[lines][2],char keygen[2][buf_size],char input[buf_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/***/
char *intostr(int value){
  int i;
  int temp=value;
  char *x;
  char y[12];
  x=(char*)calloc(12,sizeof(char));
  for(i=0;temp!=0;i++,temp/=10);
    x[i]='\0';i--;
  if(!(i<12))
    return NULL;
  for(temp=value,temp%=10;!(i<0);x[i]=temp+48,value/=10,temp=value,i--,temp%=10);
  strcpy(y,x);
  free(x);
  return y;
}


//& bitwise AND, && logical bool and
//solving problems using complicated
//methods make it more viable and strong.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
```

```
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;

  Serial.flush();
  //Serial.println(EOF,DEC);
  //INIC
  for( Hist[C]=0,Hist[B]=0; TRUE; Hist[C] = Entry[C], Hist[B] = Entry[B]){
    //ENTRADA Pinos
    Entry[C]=PINC;
    //ENTRADA Serial
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }

    if((Entry[C] == Hist[C]) && (Entry[B] == Hist[B]))
      continue;
    //if(Entry[C] != Hist[C])
    Serial.println(Entry[C],DEC);
    delay(30);
    PORTB = Entry[B];
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
```

```
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.
```