```
#define LEN(array) ((sizeof(array))/(sizeof(array[0])))

#define PIN_CLKM 13 // Clock dos botões da mesa
#define PIN_RSTM 10 // Resete dos botões da mesa
#define PIN_OUTM 11 // Sinal serial dos botões da mesa

#define PIN_CLKD 12 // Clock dos displays de senha
#define PIN_RSTD 9 // Resete dos displays de senha
#define PIN_INPDM 8 // Input dos displays da mesa
#define PIN_INPDS 7 // Input dos displays de senha
#define PIN_OUTBT 6 // Botão para mostrar as senhas

#define NUM_MESAS 8

int mesas[NUM_MESAS] = {0, 0, 0, 0, 0, 0, 0, 0};
int senhas[NUM_MESAS] = {0, 0, 0, 0, 0, 0, 0, 0};
int num_senha = 1;

#define BTN_MS 0 // Botão de mostrar as senhas ou parar de mostrar
int buttons;

// Controlar visibilidade das senhas
unsigned long int time_clock_senhas = 0, tempo_mostrar_senhas = 2000;
int index_mostrar_senhas = 0, pode_mostrar_senhas = 0;

// Controlar visibilidade das mesas
unsigned long int time_clock_mesa = 0, tempo_mostrar_mesa = 2000;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9800);
  pinMode(PIN_CLKM, OUTPUT);
  pinMode(PIN_RSTM, OUTPUT);
  pinMode(PIN_OUTM, INPUT);
  pinMode(PIN_OUTBT, INPUT);
  pinMode(PIN_CLKD, OUTPUT);
  pinMode(PIN_RSTD, OUTPUT);
  pinMode(PIN_INPDS, OUTPUT);
  pinMode(PIN_INPDM, OUTPUT);
  Serial.println("Iniciou!!");
}

void reset_register(int port) {
  digitalWrite(port, 0);
  digitalWrite(port, 1);
}

void clock_register(int port) {
  digitalWrite(port, 0);
  digitalWrite(port, 1);
}

int indexminint(int pi[], int size_pi) {
  int min_index=0;
  for (int index=1; index < size_pi; ++index) {
    if ((pi[index] < pi[min_index] && pi[index]) || !pi[min_index]) {
      min_index = index;
    }
  }
  return min_index;
}

int mostrar_senha(int index) {
  if (mesas[index] == 0) { return 0; }
  reset_register(PIN_RSTD);
  int _mesa = index+1;
  int _senha = senhas[index];
  // necessário para substituir as letras da conversão
  if (_senha > 9) { _senha += 6; }
  for (int bi=0; bi<8; bi++) {
    digitalWrite(PIN_INPDM, bitRead(_mesa, bi));
    digitalWrite(PIN_INPDS, bitRead(_senha, bi));
    clock_register(PIN_CLKD);
  }
  return 1;
}

void loop() {
  reset_register(PIN_RSTM);
  int output_serial = 0;

  // Leitura da porta serial das mesas e dos botões
  for (int index=0; index<8; index++) {
    output_serial = digitalRead(PIN_OUTM);
    if (output_serial != mesas[index]) {
      mesas[index] = output_serial;
      if (output_serial) {
        senhas[index] = num_senha;
        num_senha++;
        if (num_senha == 99) { num_senha = 0; }
      } else {
        senhas[index] = 0;
      }
    }
    bitWrite(buttons, index, digitalRead(PIN_OUTBT));
    clock_register(PIN_CLKM);
  }

  if (bitRead(buttons, BTN_MS)) {
    pode_mostrar_senhas = !pode_mostrar_senhas;
    delay(200);
  }

  if (pode_mostrar_senhas && (millis()-time_clock_senhas) >= tempo_mostrar_senhas) {
    time_clock_senhas = millis();
    if (index_mostrar_senhas == NUM_MESAS) {
      index_mostrar_senhas = 0;
      pode_mostrar_senhas = 0;
    }
    else {
      while (!senhas[index_mostrar_senhas] && (index_mostrar_senhas < NUM_MESAS-1)) {
        index_mostrar_senhas++;
      }
      mostrar_senha(index_mostrar_senhas);
      index_mostrar_senhas++;
    }
  }
  else if (!pode_mostrar_senhas && (millis()-time_clock_mesa) >= tempo_mostrar_mesa) {
    time_clock_mesa = millis();
    int index_menor_senha = indexminint(senhas, LEN(senhas));
    mostrar_senha(index_menor_senha);
  }
  delay(250);
}
```