This ULU

The *ULU.06 Dual JK flip-flop* provides two JK flip-flops. These can be used is several ways, as described further in this datasheet.

Used parts

Only standard parts are used:

1x casing 80 x 50 x 20mm;
10x 1-bit signal connector;
10x black O-ring 9 x 5 x 2mm;
1x power socket;
1x Arduino Nano;
6x 10K pull-down resistor;
2x 3mm LED;
2x resistor to dim the LEDs;
2x LED holder;
2x micro (G6K-2F-Y-5VDC) relay;
2x fly back diode (1N4148);
2x 1-pole ON-OFF-ON switch;
1x 11 x 8-hole experiment PCB

Construction

The standard ULU specifications are applicable and given in the datasheet *ULU.00 – Common specifications*.

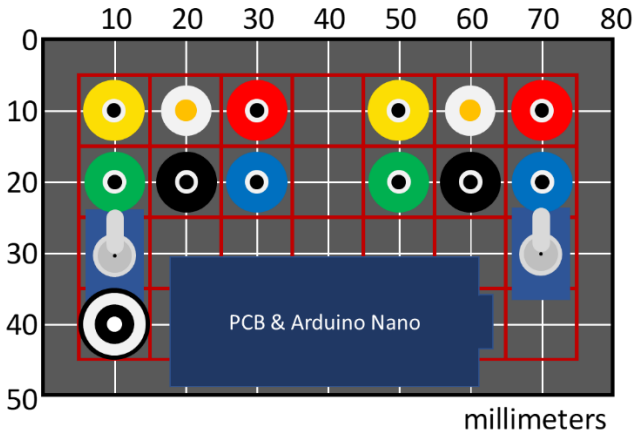

Figure 1 – Drill guide

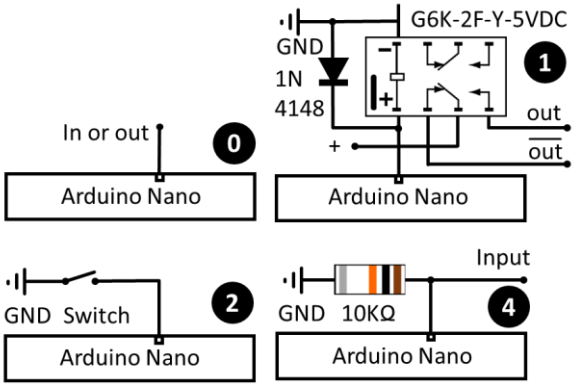

Figure 2 – Arduino interfaces

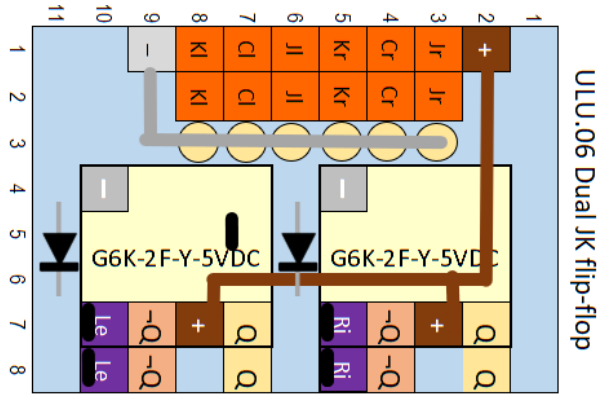

Figure 3 – Layout relay & resistor PCB

Figure 4 – Finished ULU

	Port	Con.	Rest.	Func.	Interface	Signal
1.	D2			I	4	Jright
2.	D3			I	4	Cright
3.	D4			I	4	Kright
4.	D5			T	2	Kright = $\overline{\text{Jright}}$
5.	D6			O	1	Qright
6.	D7			I	4	Jleft
7.	D8			I	4	Cleft
8.	D9			I	4	Kleft
9.	D10			T	2	Kleft = $\overline{\text{Jleft}}$
10.	D11			O	1	Qleft
11.	D13		O	L		Hartbeat LED
12.	A0			I	2	OneSocketIleft
13.	A1			I	2	OneSocketright
14.	+5V		I	I	0	DC +5V
15.	GND		I	I	0	GND

Input, Output, Led, SPI, Toggle switch, Rotary switch

Figure 5 – Pinout Arduino Nano

Trigger	Inputs		Output						Inference
			Present State		Intermediate		Next State		
CLK	J	K	Q	$\bar{Q}$	M_1	M_2	Q	$\bar{Q}$	
↑	0	0	0	1	0	1	Latched		No Change
↓			0	1	Latched		0	1	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	0	1	0	1	0	1	Latched		Reset
↓			0	1	Latched		0	1	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	
↑	1	0	0	1	1	0	Latched		Set
↓			0	1	Latched		1	0	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	1	1	0	1	1	0	Latched		Toggles
↓			0	1	Latched		1	0	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	

Figure 6 – Truth table JK flip-flop

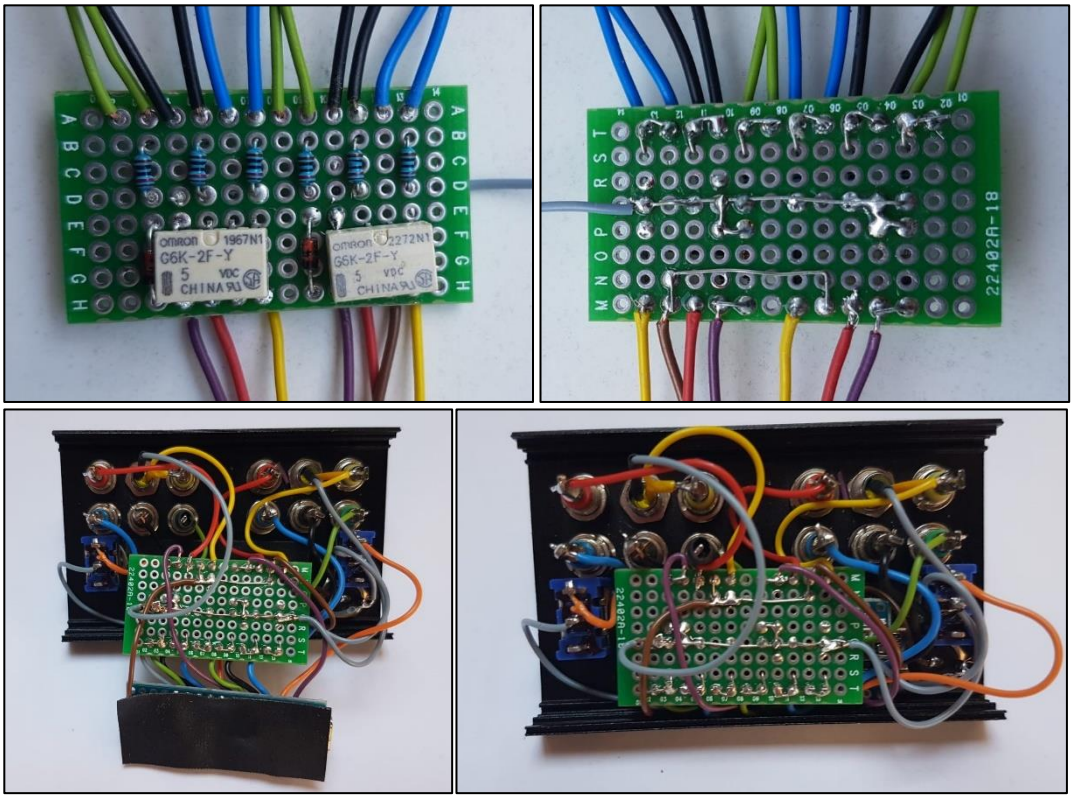

Figure 7 – Various states of PCB fabrication

First the PCB is cut to size, then all the resistors, diodes and wires are soldered. Input wires are soldered on the top side, output (relay) wires on the bottom side. After that, the wires for the Arduino are cut to

length and soldered to the Arduino board. The Arduino Nano pinout is given in Figure 5. After that, the relays can be easily soldered on the top PCB side and will connect to the wires on the bottom side.

Make sure the top of PCB and the top of the Arduino board can be folded towards each other. A piece of tape on the Arduino bottom side will prevent unwanted electrical contacts. This is shown in Figure 7

Both LEDs have their (long) + pin left intact and soldered them to the (yellow) output connector. The resistor is soldered to the (short) – pin. The common of both switches are connected to the ground (-) and the contact that makes the switch on when the rocker is set downwards, is connected to the Arduino.

If there is a risk that the PCB makes unwanted contact with the connectors, put some heat shrink tube on those connectors.

Usage

The JK flip-flop has three inputs and one output Q. The other output $\bar{Q}$ is the inverse of Q (NOT Q). The basic layout of the JK flip flop is drawn in Figure 8. This ULU can be used in three ways:

1. Set/reset flip-flop

Set: when J and clock are set to 1 simultaneously, the output Q will become 1 and $\bar{Q}$ will become 0.
Reset/clear: when K and clock are set to 1 simultaneously, the output Q will become 0 and $\bar{Q}$ 1.

This type of flip-flop can be used to 'remember' the input. See Figure 9.

2. Toggle flip-flop

When J, K and clock are set to 1 simultaneously, the output Q will toggle. If it was 0, it will become 1 and if it was 1 it will become 0.

This type of flip-flop can be used to make a counter. See Figure 10.

3. Memory/data storage

When K is set to the inverse of J (Not J) the flip-flop can used to store data. When the data line is attached to the K and the clock is set to 1, the flip-flop will remember the offered data value. To facilitate this type of flip-flop, a switch is provided. When the switch is set to the top, the K input is kept equal to the inverse of J.

This type of flip-flop can be used as a one-bit register or a shift register. See Figure 11.

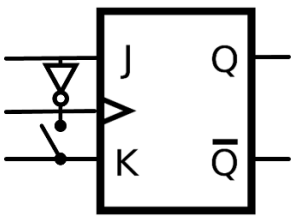

Figure 8 – The JK flip-flop

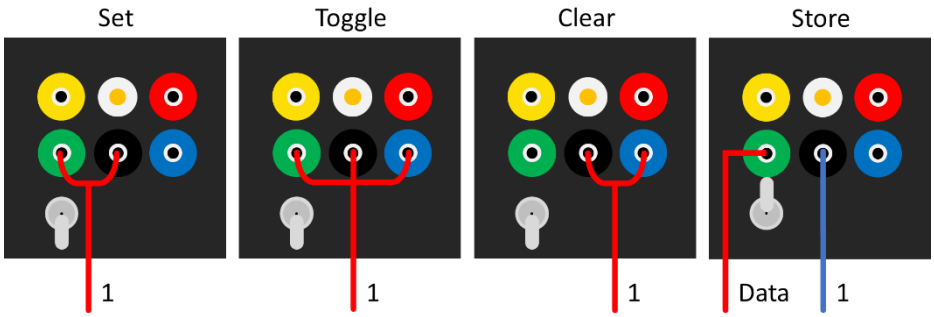

Figure 9 – Using the JK flip-flop

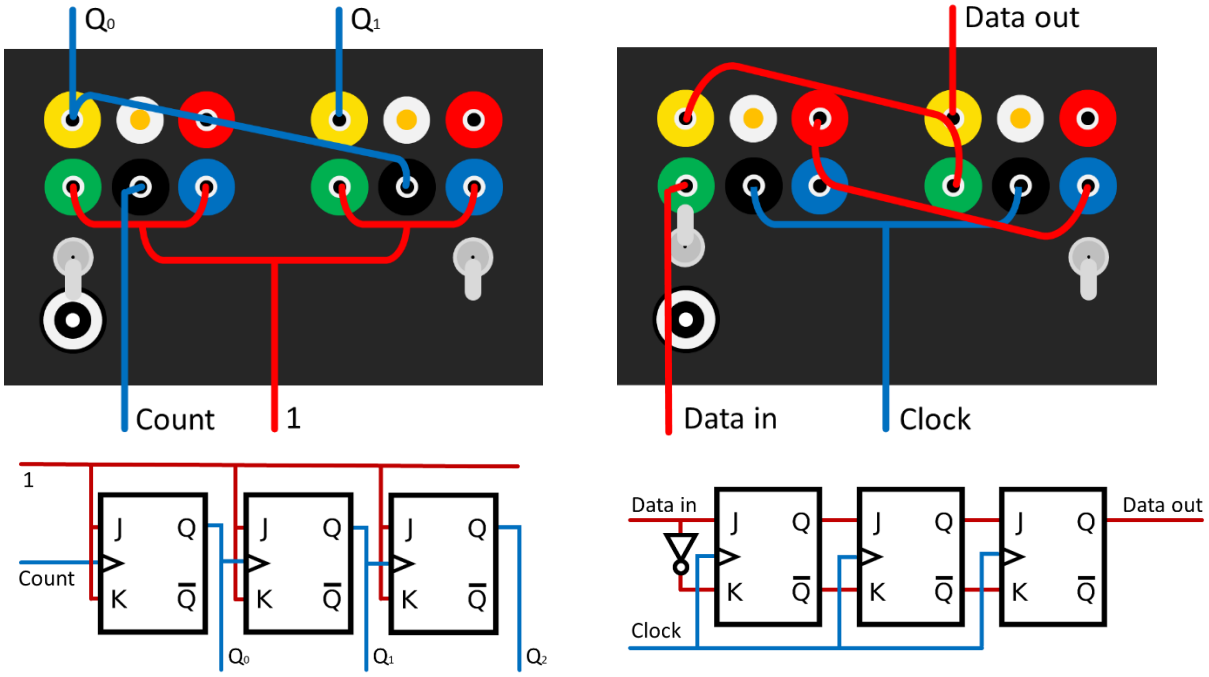

Figure 10 – Counter

Figure 11 – Shift register

For easy use of this ULU there is a single socket modus. For that, the switch need to be in the center position. This modus works exactly as the JK flip-flop modus, apart from the fact that only one line is used for controlling the flip-flop. The way to control the flip-flop in this modus is shown in Figure 12.

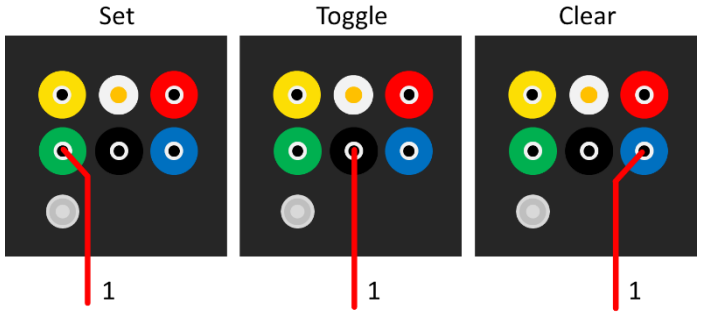

Figure 12 – Single socket control

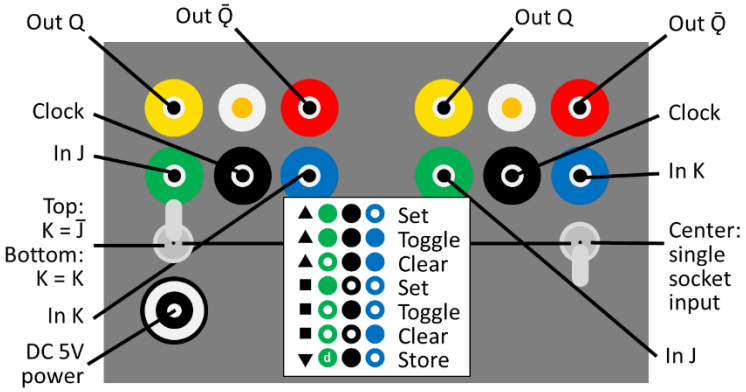

Figure 13 – Connectors & controls

Arduino Nano solder check

```
/* ULU.06  Dual JK flip-flop - solder check  */
/* CC BY-NC-SA Jeroen Brinkman */

int jleft, cleft, kleft, uleft, dleft; // left variables
int jright, cright, kright, uright, dright; // right variables

#define WAIT 150
#define RJ 2
#define RC 3
#define RK 4
#define RU 5
#define RQ 6
#define LJ 7
#define LC 8
#define LK 9
#define LU 10
#define LQ 11
#define LD A0
#define RD A1
#define HEART  13

void setup() {
   Serial.begin(9600); // enables the serial monitor
   /* Define I/O pins */
   pinMode(RU,  INPUT_PULLUP);
   pinMode(RD,  INPUT_PULLUP);
   pinMode(RJ,  INPUT);
   pinMode(RC,  INPUT);
   pinMode(RK,  INPUT);
   pinMode(RQ,  OUTPUT);
   pinMode(LU,  INPUT_PULLUP);
   pinMode(LD,  INPUT_PULLUP);
   pinMode(LJ,  INPUT);
   pinMode(LC,  INPUT);
   pinMode(LK,  INPUT);
   pinMode(LQ,  OUTPUT);
   pinMode(HEART, OUTPUT); // blinking led showing the program hartbeat

   /* Every ULU with an Arduino introduces itself. This one uses blinking leds */
   digitalWrite(LQ, HIGH); delay(WAIT);
   digitalWrite(LQ, LOW); digitalWrite(RQ, HIGH); delay(WAIT);
   digitalWrite(LQ, HIGH); digitalWrite(RQ, LOW); delay(WAIT);
   digitalWrite(LQ, LOW); digitalWrite(RQ, HIGH); delay(WAIT);
   digitalWrite(LQ, HIGH); digitalWrite(RQ, LOW); delay(WAIT);
   digitalWrite(LQ, LOW); digitalWrite(RQ, HIGH); delay(WAIT); digitalWrite(RQ, LOW);
}
void blink(int side, int show){
  for (int counter=1; counter <= show; counter = counter+1){
    digitalWrite(side, HIGH);
    delay(WAIT);
    digitalWrite(side, LOW);
    delay(WAIT);
  }
  delay(WAIT*5);
}

void loop(){
   digitalWrite(HEART, (millis() / 1000) % 2); //1s hartbeat for the onbord led
   uright = !digitalRead(RU); if(uright == 1) blink(RQ, 1); Serial.print((uright == 1) ? "RU " : "   ");
   dright = !digitalRead(RD); if(dright == 1) blink(RQ, 2);  Serial.print((dright == 1) ? "RD " : "   ");
   jright = digitalRead(RJ); if(jright == 1) blink(RQ, 3); Serial.print((jright == 1) ? "RJ " : "   ");
   cright = digitalRead(RC); if(cright == 1) blink(RQ, 4); Serial.print((cright == 1) ? "RC " : "   ");
   kright = digitalRead(RK); if(kright == 1) blink(RQ, 5); Serial.print((kright == 1) ? "RK " : "   ");

   uleft = !digitalRead(LU); if(uleft == 1) blink(LQ, 1); Serial.print((uleft == 1) ? "LU " : "   ");
   dleft = !digitalRead(LD); if(dleft == 1) blink(LQ, 2); Serial.print((dleft == 1) ? "LD " : "   ");
   jleft = digitalRead(LJ); if(jleft == 1) blink(LQ, 3); Serial.print((jleft == 1) ? "LJ " : "   ");
   cleft = digitalRead(LC); if(cleft == 1) blink(LQ, 4); Serial.print((cleft == 1) ? "LC " : "   ");
   kleft = digitalRead(LK); if(kleft == 1) blink(LQ, 5); Serial.println((kleft == 1) ? "LK " : "   ");
}
```

Arduino Nano program

```
/* ULU.06  Dual JK flip-flop - program code  */
/* CC BY-NC-SA Jeroen Brinkman */

bool triggerleft, triggerright; // booleans
int i;
int jleft, lastjleft, cleft, lastcleft, kleft, lastkleft, setleft, qleft, switchleft; // left variables
int jright, lastjright, cright, lastcright, kright, lastkright, setright, qright, switchright; // right variables

#define BOUNCE 8
#define WAIT 150
#define RJ 2
#define RC 3
#define RK 4
#define RU 5
#define RQ 6
#define LJ 7
#define LC 8
#define LK 9
#define LU 10
#define LQ 11
#define LD A0
```

```
#define RD A1
#define HEART  13

void setup() {

   /* Define I/O pins */
   pinMode(RU,  INPUT_PULLUP);
   pinMode(RD,  INPUT_PULLUP);
   pinMode(RJ,  INPUT);
   pinMode(RC,  INPUT);
   pinMode(RK,  INPUT);
   pinMode(RQ,  OUTPUT);
   pinMode(LU,  INPUT_PULLUP);
   pinMode(LD,  INPUT_PULLUP);
   pinMode(LJ,  INPUT);
   pinMode(LC,  INPUT);
   pinMode(LK,  INPUT);
   pinMode(LQ,  OUTPUT);
   pinMode(HEART, OUTPUT); // blinking led showing the program hartbeat

   /* Every ULU with an Arduino introduces itself. This one uses blinking leds */
   digitalWrite(LQ, HIGH); delay(WAIT);
   digitalWrite(LQ, LOW); digitalWrite(RQ, HIGH); delay(WAIT);
   digitalWrite(LQ, HIGH); digitalWrite(RQ, LOW); delay(WAIT);
   digitalWrite(LQ, LOW); digitalWrite(RQ, HIGH); delay(WAIT);
   digitalWrite(LQ, HIGH); digitalWrite(RQ, LOW); delay(WAIT);
   digitalWrite(LQ, LOW); digitalWrite(RQ, HIGH); delay(WAIT); digitalWrite(RQ, LOW);

   /* Initialize variables */
   triggerright = false; setright = 0; qright = 0;
   lastcright = 0; lastjright = 0; lastkright = 0;
   triggerleft = false; setleft = 0; qleft = 0;
   lastcleft = 0; lastjleft = 0; lastkleft = 0;
}

void loop(){
   digitalWrite(HEART, (millis() / 1000) % 2); //1s hartbeat for the onbord led

   /* read the switches */
   switchright =  !digitalRead(RU) + (!digitalRead(RD) * 2);
   switchleft =  !digitalRead(LU) + (!digitalRead(LD) * 2);

   /* read the input */
   jright = 0; cright = 0; kright = 0;
   jleft = 0; cleft = 0; kleft = 0;
   for (i = 0; i < BOUNCE; i++) { //debounce routine
      jright += digitalRead(RJ);
      cright += digitalRead(RC);
      kright += digitalRead(RK);
      jleft += digitalRead(LJ);
      cleft += digitalRead(LC);
      kleft += digitalRead(LK);
      delay(1);
   }
   jright = (jright > i / 2) ? 1 : 0;
   cright = (cright > i / 2) ? 1 : 0;
   kright = (kright > i / 2) ? 1 : 0;
   jleft = (jleft > i / 2) ? 1 : 0;
   cleft = (cleft > i / 2) ? 1 : 0;
   kleft = (kleft > i / 2) ? 1 : 0;

   /* Determine right trigger */
   triggerright = ((cright == 0) && (lastcright == 1));
   if ((switchright == 0) &&  (lastjright == 1) && (jright == 0)) triggerright = true;
   if ((switchright == 0) &&  (lastkright == 1) && (kright == 0)) triggerright = true;

   /* Determine & execute right action */
   if (triggerright ){
      triggerright = false;
      switch (switchright) {
         case 0: // Single socket input
            if (lastjright == 1) qright = 1;
            if (lastcright == 1) qright = 0;
            if (lastkright == 1) qright = ((qright == 0) ? 1 : 0);
            break;
         case 1: // Store J
            qright =  lastjright;
            break;
         case 2: // Regular use
            if ((lastjright == 1) && (lastkright == 0)) qright = 1;
            if ((lastkright == 1) && (lastjright == 0)) qright = 0;
            if ((lastjright == 1) && (lastkright == 1)) qright = ((qright == 0) ? 1 : 0);
            break;
      }
      digitalWrite(RQ, (qright == 1 ) ? HIGH : LOW);
   }
   lastjright = jright; lastcright = cright; lastkright = kright;

   /* Determine left trigger */
   triggerleft = ((cleft == 0) && (lastcleft == 1));
   if ((switchleft == 0) &&  (lastjleft == 1) && (jleft == 0)) triggerleft = true;
   if ((switchleft == 0) &&  (lastkleft == 1) && (kleft == 0)) triggerleft = true;

   /* Determine & execute left action */
   if (triggerleft ){
      triggerleft = false;
      switch (switchleft) {
         case 0: // Single socket input
            if (lastjleft == 1) qleft = 1;
            if (lastcleft == 1) qleft = 0;
```

```
            if (lastkleft == 1) qleft = ((qleft == 0) ? 1 : 0);
            break;
         case 1: // Store J
            qleft =  lastjleft;
            break;
         case 2: // Regular use
            if ((lastjleft == 1) && (lastkleft == 0)) qleft = 1;
            if ((lastkleft == 1) && (lastjleft == 0)) qleft = 0;
            if ((lastjleft == 1) && (lastkleft == 1)) qleft = ((qleft == 0) ? 1 : 0);
            break;
      }
   digitalWrite(LQ, (qleft == 1 ) ? HIGH : LOW);
   }
   lastjleft = jleft; lastcleft = cleft; lastkleft = kleft;
}
```