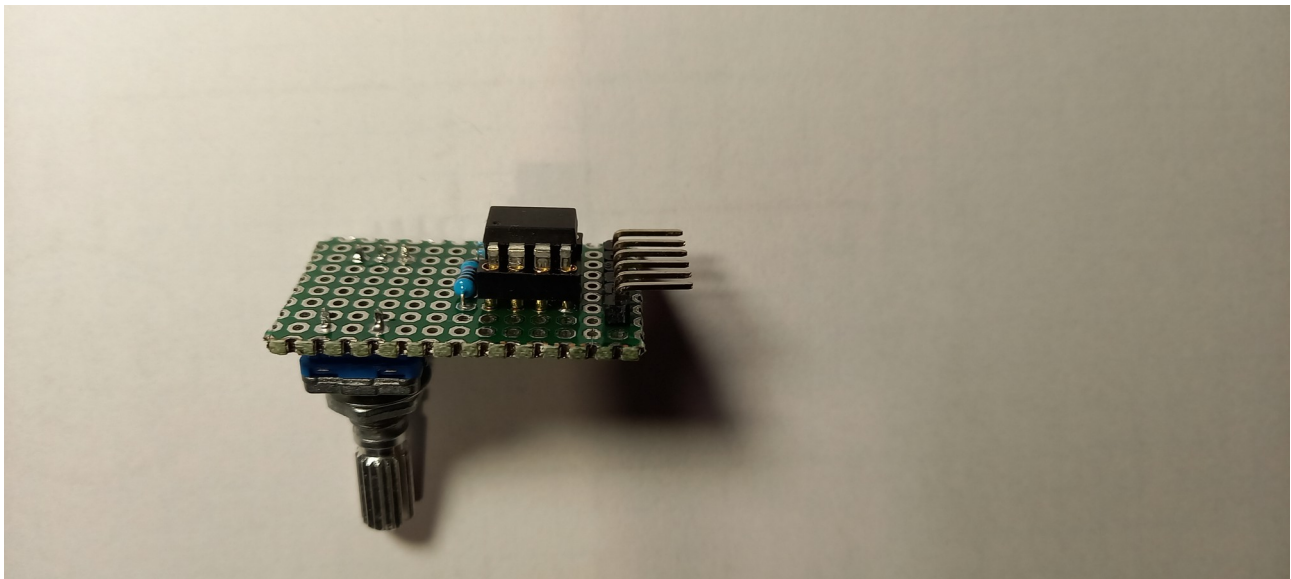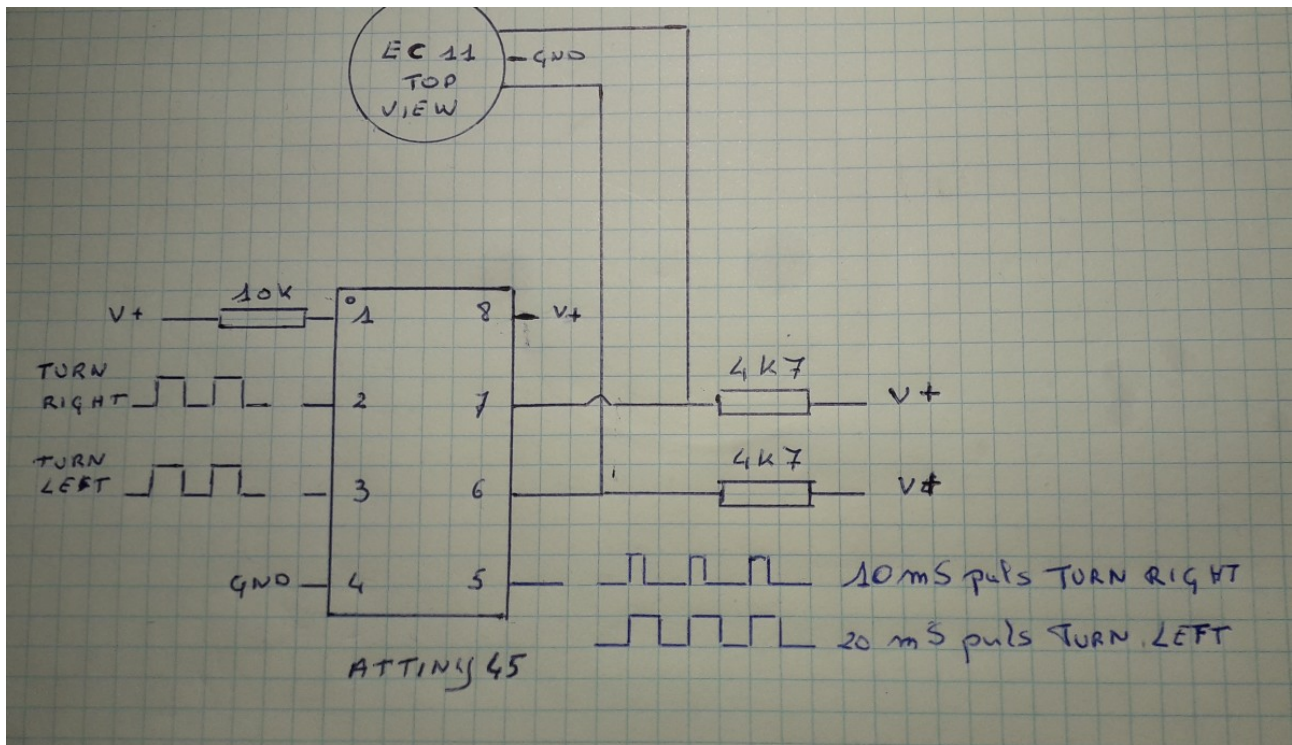# EC11 rotary encoder the easy way
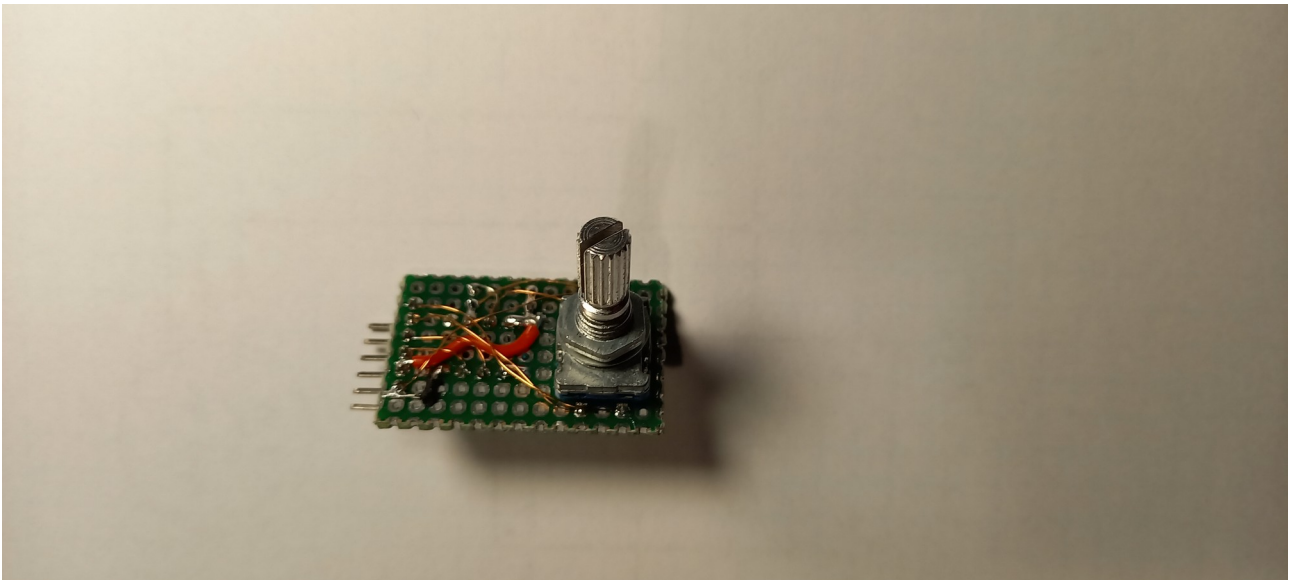
If desired, only 1 input on main MCU per rotary encoder, even with push button

Specifications

power supply 3.3V or 5V
no extra capacitors or resistors
no libraries
works for slow and fast rotation
no unwanted pulses
cheap about 3€
easy to build

Only as a rotary encoder (ATTINY45)

what do we need
1x   ATTINY45
2x   4K7
1x   10K
1x   EC11

We have 2 options for the connection to the main MCU

with 2 outputs
1x pulse output when turning right
1x pulse output when turning left
The ATTINY45 provides 20 millisecond pulses on either the right turn output or the left turn output.
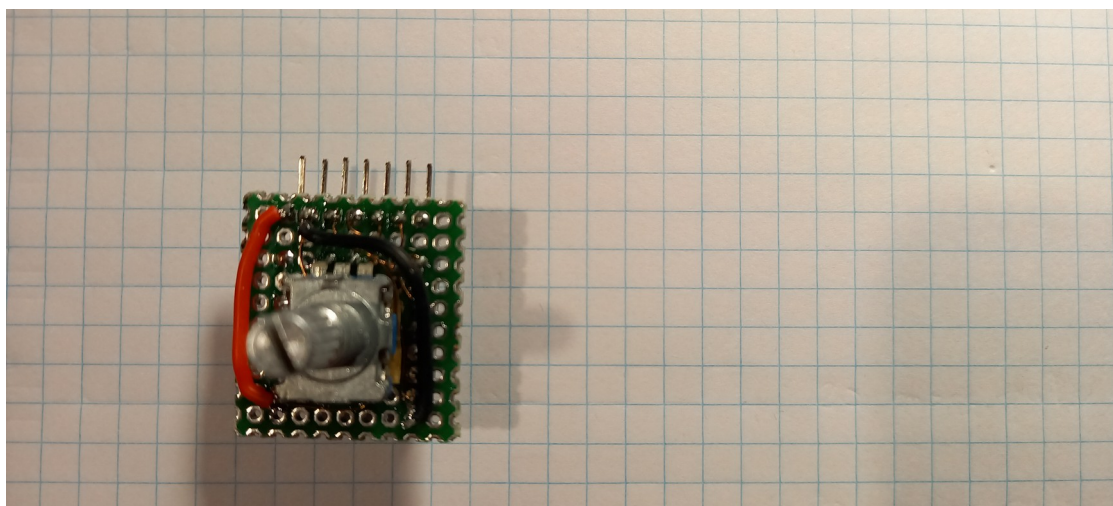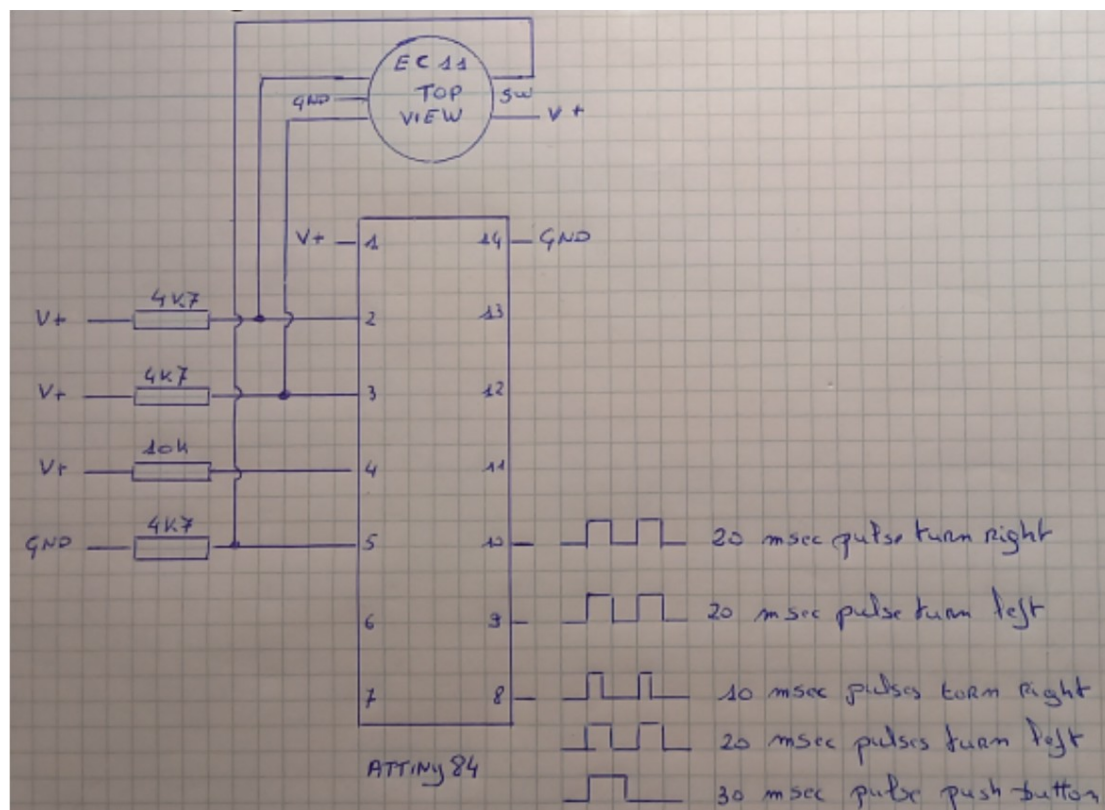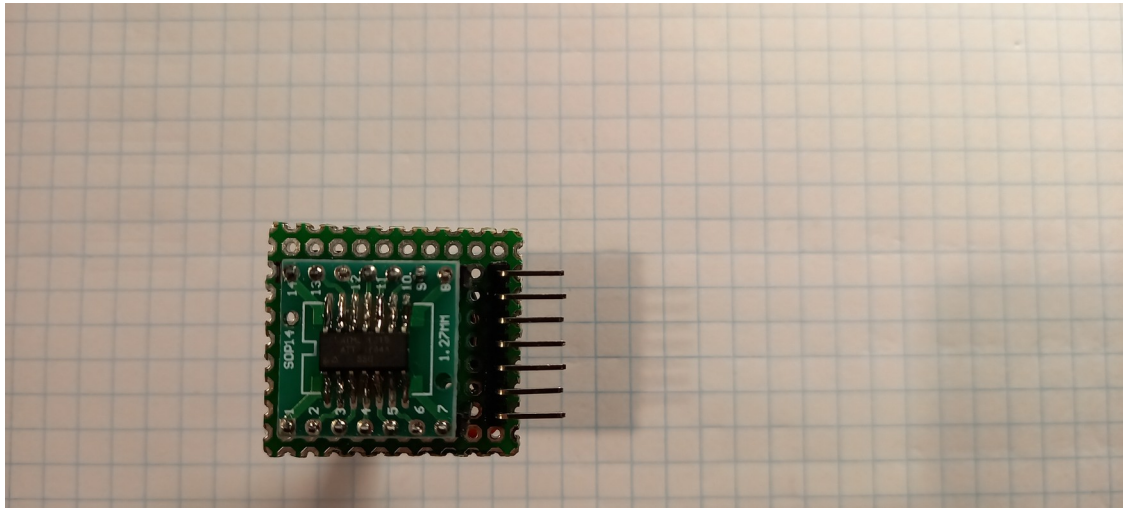
with 1 output
1x output
The ATTINY45 delivers
10 millisecond pulses when turning clockwise
20 millisecond pulses when turning counterclockwise on the same output.
decoding must be done on main mcu
ESP32 example program at the end of this pdf file

Thanks to
https://www.elektormagazine.nl/labs/rotary-encoder-on-a-single-mcu-pin
for the 1 input idea

As a rotary encoder with push button (ATTINY84)

what do we need
1x   ATTINY84
3x   4K7
1x   10K
1x   EC11 with pushbutton

We have 2 options for the connection to the main MCU

with 2 outputs
1x pulse output when turning right
1x pulse output when turning left
The ATTINY84 provides 20 millisecond pulses on either the right turn output or the left turn output.
Push button must be connected separately to main MCU

with 1 output
The ATTINY84 delivers
10 milliseconds pulses when turning clockwise
20 milliseconds pulses when turning counterclockwise
30 milliseconds pulse when push button is pressed
and this on the same digital output
decoding must be done on main mcu
ESP32 example program at the end of this pdf file

Thanks to
https://www.elektormagazine.nl/labs/rotary-encoder-on-a-single-mcu-pin
for the 1 input idea

When programming the ATTINY, do not forget to program the boat loader

ATTINY45 program (only rotary encoder)

```
/*
* MIT License
*
* Copyright (c) 2023 thieu-b55
*
* Permission is hereby granted, free of charge, to any person obtaining a copy
* of this software and associated documentation files (the "Software"), to deal
* in the Software without restriction, including without limitation the rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in all
* copies or substantial portions of the Software.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*/

#define A_INPUT   1
#define B_INPUT   2
#define R_UIT     3
#define L_UIT     4
#define L_R_UIT   0

bool input_1_bool;
bool input_2_bool;
bool input_1_vorig_bool;
unsigned long begin_millis;

void setup() {
  pinMode(A_INPUT, INPUT);
  pinMode(B_INPUT, INPUT);
  pinMode(R_UIT, OUTPUT);
  pinMode(L_UIT, OUTPUT);
  pinMode(L_R_UIT, OUTPUT);
}

void loop() {
  input_1_bool = digitalRead(A_INPUT);
  input_2_bool = digitalRead(B_INPUT);
  delay(2);
  if(input_1_bool == digitalRead(A_INPUT)){
    if(input_1_bool != input_1_vorig_bool){
      if((!input_1_bool) && (!input_2_bool)){
        begin_millis = millis();
        digitalWrite(R_UIT, true);
        digitalWrite(L_R_UIT, true);
        while((millis() - begin_millis) < 10){
        }
        digitalWrite(L_R_UIT, false);
        while((millis() - begin_millis) < 20){
        }
        digitalWrite(R_UIT, false);
        while((millis() - begin_millis) < 38){
        }
      }
      if((!input_1_bool) && (input_2_bool)){
        begin_millis = millis();
        digitalWrite(L_UIT, true);
        digitalWrite(L_R_UIT, true);
        while((millis() - begin_millis) < 20){
```

```
      }
      digitalWrite(L_UIT, false);
      digitalWrite(L_R_UIT, false);
      while((millis() - begin_millis) < 38){
      }
    }
    input_1_vorig_bool = input_1_bool;
  }
 }
}
```

ATTINY84 program (rotary encoder with push button)

```
/*
 * ATTINY84
 */

#define A_INPUT   10
#define B_INPUT    9
#define SWITCH     8
#define R_UIT      3
#define L_UIT      4
#define MILLIS_UIT 5

bool input_1_bool;
bool input_2_bool;
bool input_1_vorig_bool;
bool input_sw_bool;
bool input_sw_vorig_bool;
unsigned long begin_millis;

void setup() {
  pinMode(A_INPUT, INPUT);
  pinMode(B_INPUT, INPUT);
  pinMode(SWITCH, INPUT);
  pinMode(R_UIT, OUTPUT);
  pinMode(L_UIT, OUTPUT);
  pinMode(MILLIS_UIT, OUTPUT);
}

void loop() {
  input_1_bool = digitalRead(A_INPUT);
  input_2_bool = digitalRead(B_INPUT);
  delay(2);
  if(input_1_bool == digitalRead(A_INPUT)){
    if(input_1_bool != input_1_vorig_bool){
      if((!input_1_bool) && (!input_2_bool)){
        begin_millis = millis();
        digitalWrite(R_UIT, true);
        digitalWrite(MILLIS_UIT, true);
        while((millis() - begin_millis) < 10){
        }
        digitalWrite(MILLIS_UIT, false);
        while((millis() - begin_millis) < 20){
        }
        digitalWrite(R_UIT, false);
```

```
        while((millis() - begin_millis) < 38){
         }
        }
       if((!input_1_bool) && (input_2_bool)){
        begin_millis = millis();
        digitalWrite(L_UIT, true);
        digitalWrite(MILLIS_UIT, true);
        while((millis() - begin_millis) < 20){
         }
        digitalWrite(L_UIT, false);
        digitalWrite(MILLIS_UIT, false);
        while((millis() - begin_millis) < 38){
         }
        }
       input_1_vorig_bool = input_1_bool;
      }
    }
   input_sw_bool = digitalRead(SWITCH);
   delay(2);
   if(input_sw_bool == digitalRead(SWITCH)){
    if(input_sw_bool != input_sw_vorig_bool){
      if(input_sw_bool){
       digitalWrite(MILLIS_UIT, true);
       delay(30);
       digitalWrite(MILLIS_UIT, false);
       delay(8);
      }
      input_sw_vorig_bool = input_sw_bool;
     }
   }
  }
```

ESP32 test program 2 inputs

Connect IC pin 2 of ATTiny45 to GPIO26 of ESP32
Connect IC pin 3 of ATTiny45 to GPIO27 of ESP32

```
#define PLUS_INPUT  26
#define MIN_INPUT   27

int totaal_int = 0;
bool int_bool = false;

void setup() {
  Serial.begin(230400);
  pinMode(PLUS_INPUT, INPUT);
  pinMode(MIN_INPUT, INPUT);
  attachInterrupt(PLUS_INPUT, totaal_plus, RISING);
  attachInterrupt(MIN_INPUT, totaal_min, RISING);
}

void loop() {
  if(int_bool){
    Serial.println(totaal_int);
    int_bool = false;
  }
}

void totaal_plus(){
  totaal_int ++;
  int_bool = true;
}

void totaal_min(){
  totaal_int --;
  int_bool = true;
}
```

ESP32 test program 1 input

Connect
IC pin 5 ATTINY45 to GPIO26 ESP32
or
IC pin 8 ATTINY84 to GPIO26 ESP32

```
#define   PLUS_MIN_INPUT  26

int totaal_int = 0;
int verschil_millis;
bool int_bool = false;
bool switch_bool = false;
bool tijd_gestart_bool = false;
unsigned long isr_begin;


void setup() {
  Serial.begin(230400);
  pinMode(PLUS_MIN_INPUT, INPUT);
  attachInterrupt(PLUS_MIN_INPUT, totaal_plus_min, CHANGE);
}

void loop() {
  if(int_bool){
    Serial.println(totaal_int);
    int_bool = false;
  }
  if(switch_bool){
    Serial.println("switch");
    switch_bool = false;
  }
}

void totaal_plus_min(){
  if(digitalRead(PLUS_MIN_INPUT)){
    isr_begin = millis();
    tijd_gestart_bool = true;
  }
  else{
    if(tijd_gestart_bool){
      verschil_millis = millis() - isr_begin;
      if( verschil_millis < 15){
        totaal_int ++;
        int_bool = true;
      }
      if((verschil_millis > 15) && (verschil_millis < 25)){
        totaal_int --;
        int_bool = true;
      }
      if(verschil_millis > 25){
        switch_bool = true;
      }
      tijd_gestart_bool = false;
    }
  }
}
```

github
https://github.com/thieu-b55/EC11-rotary-encoder-the-easy-way

have fun,
thieu