CEC Continuing Education Center

DesignNews

C++ Primer for Embedded Systems

Day 5:

Arduino Libraries: C or C++?

Sponsored by

DigiKey

informa markets

Webinar Logistics

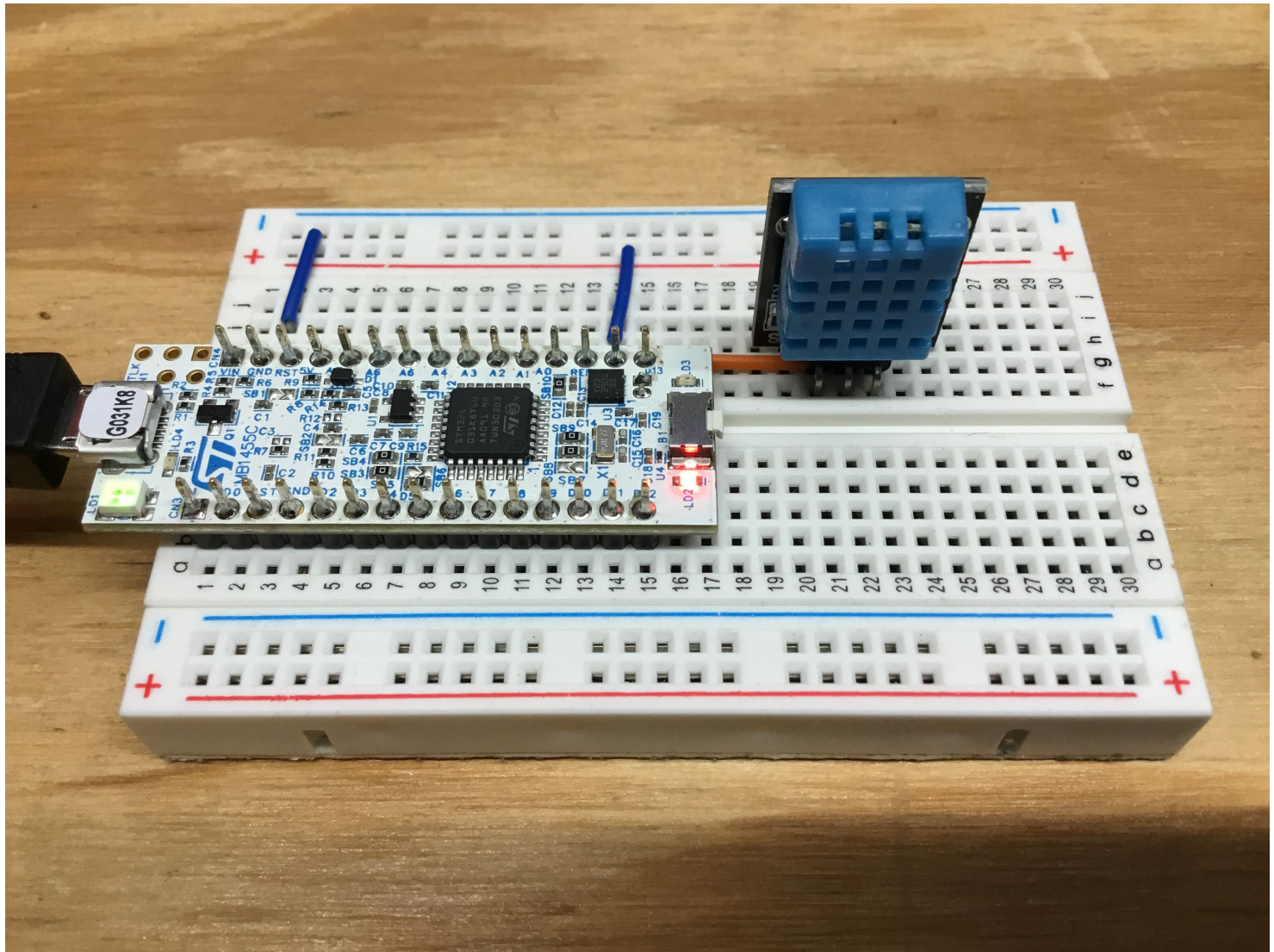
- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Fred Eady

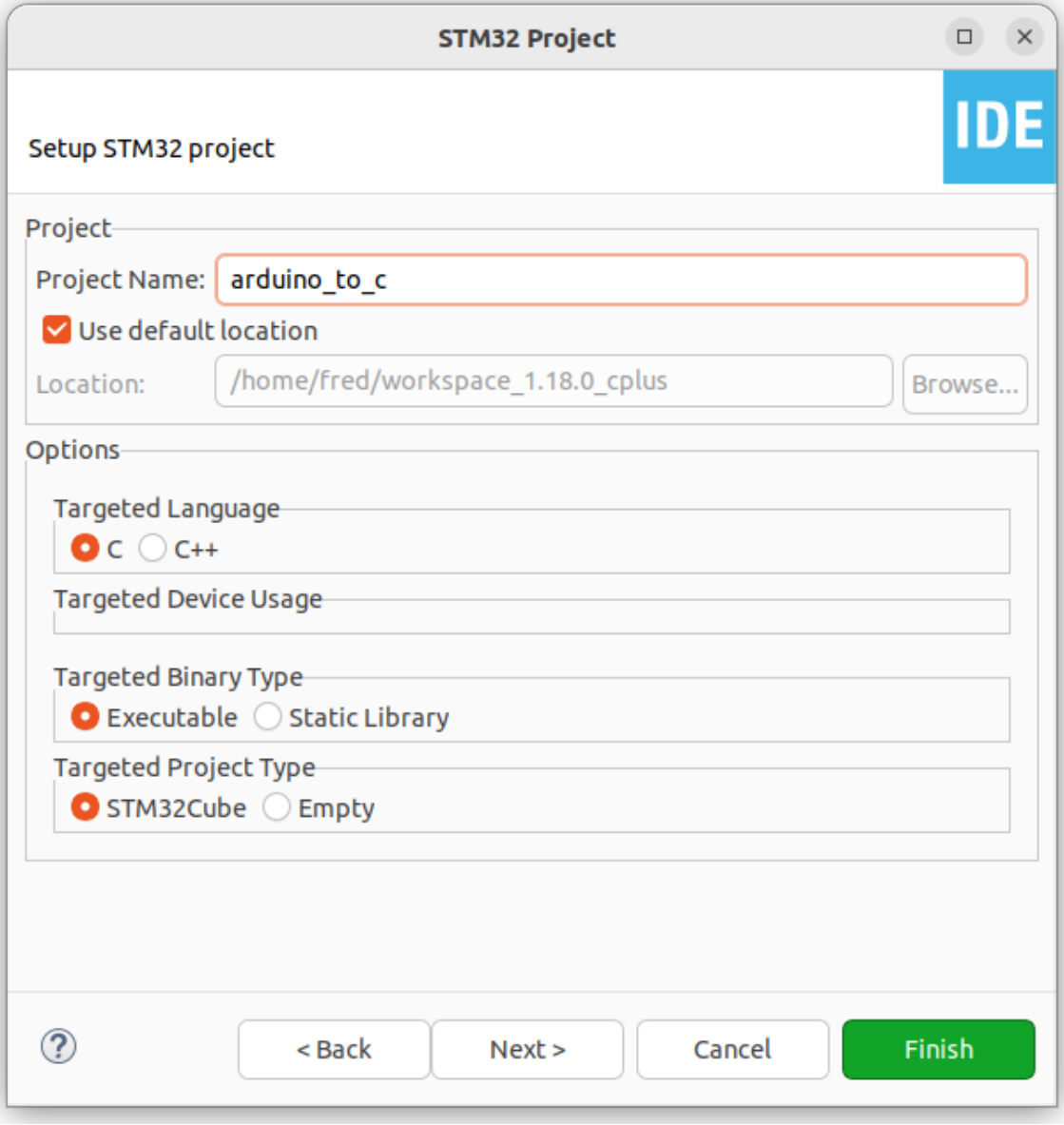
Visit ‘Lecturer Profile’ in your console for more details.

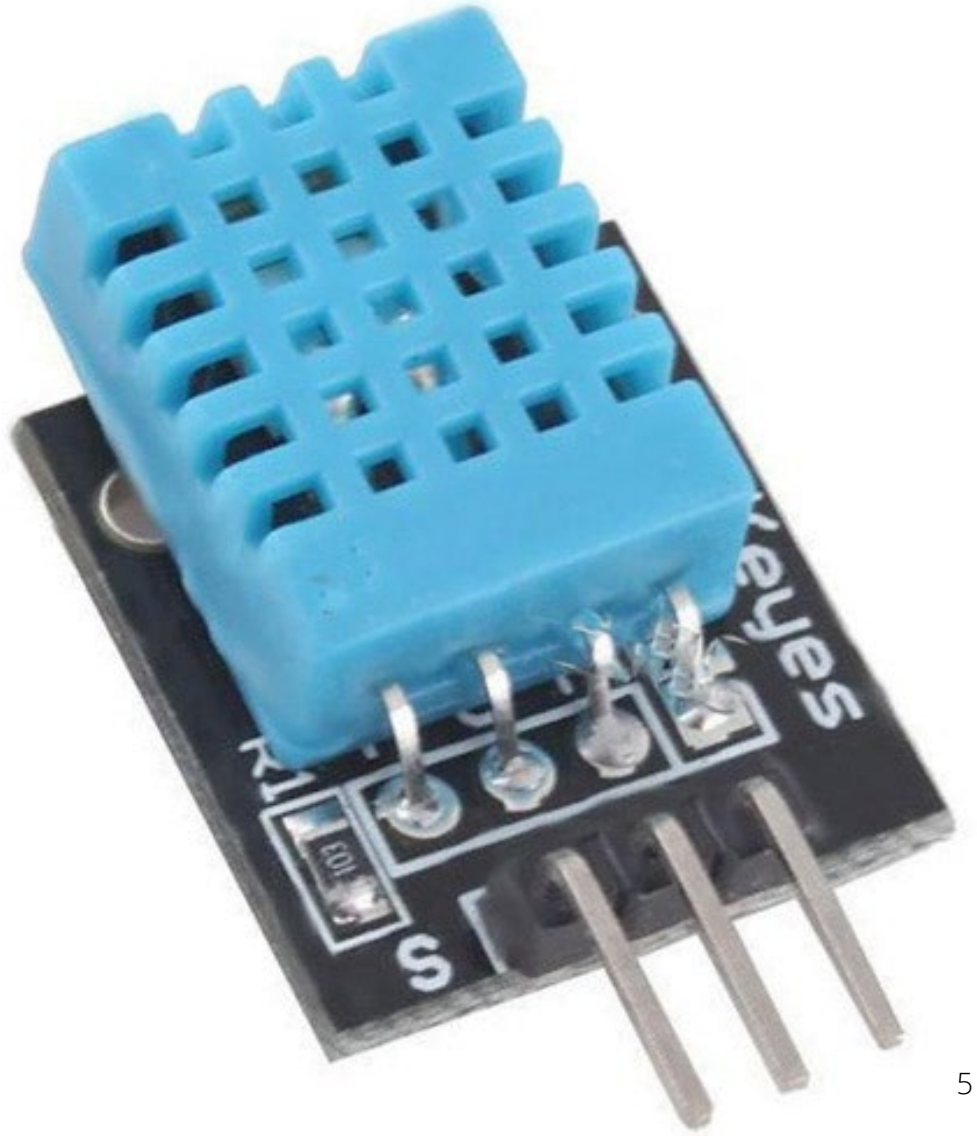
AGENDA

- **Create an STMCubeIDE C Android-to-C Project**

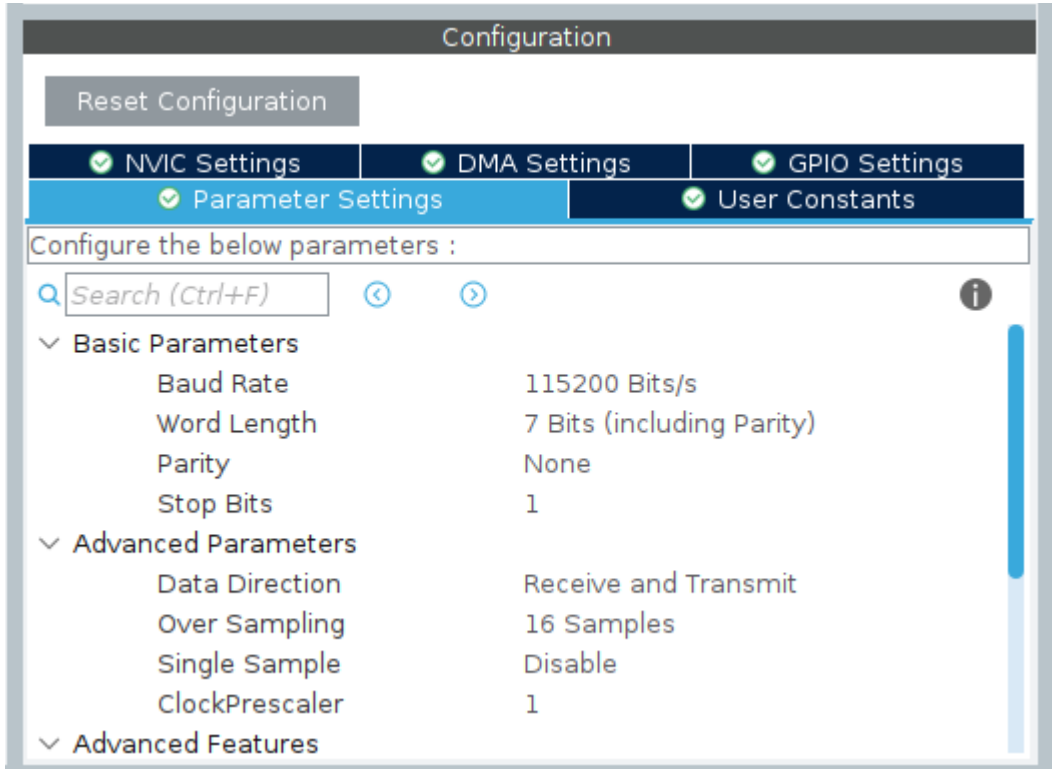


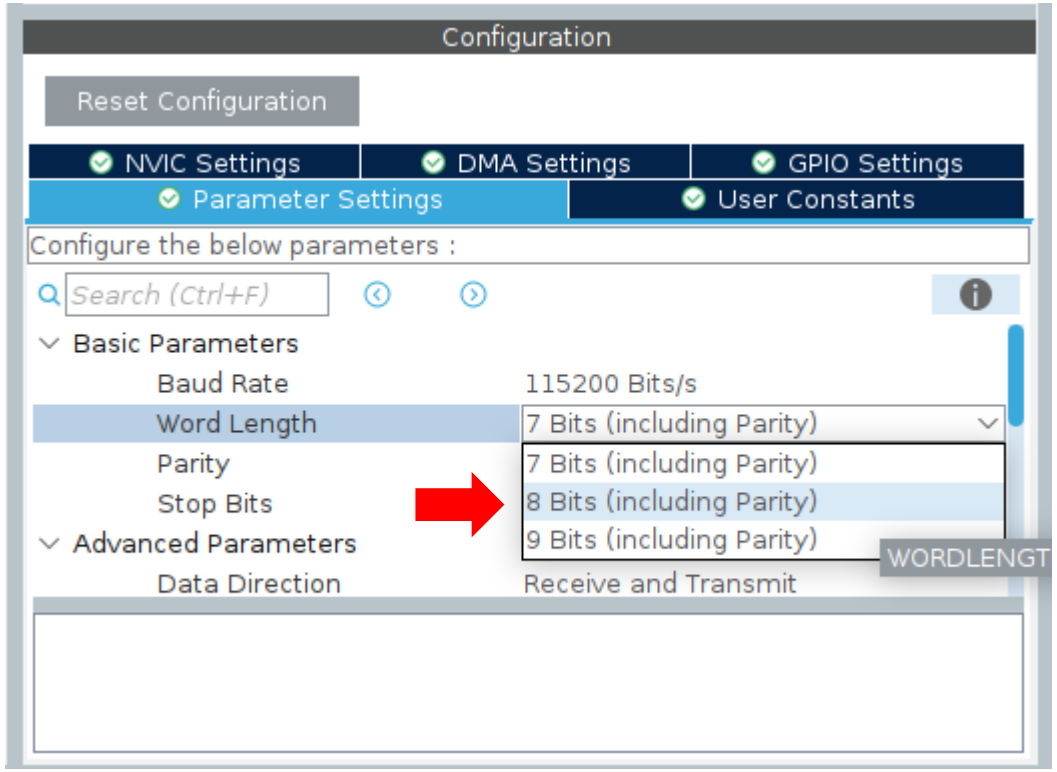
Create a New STM32CubeIDE C Project



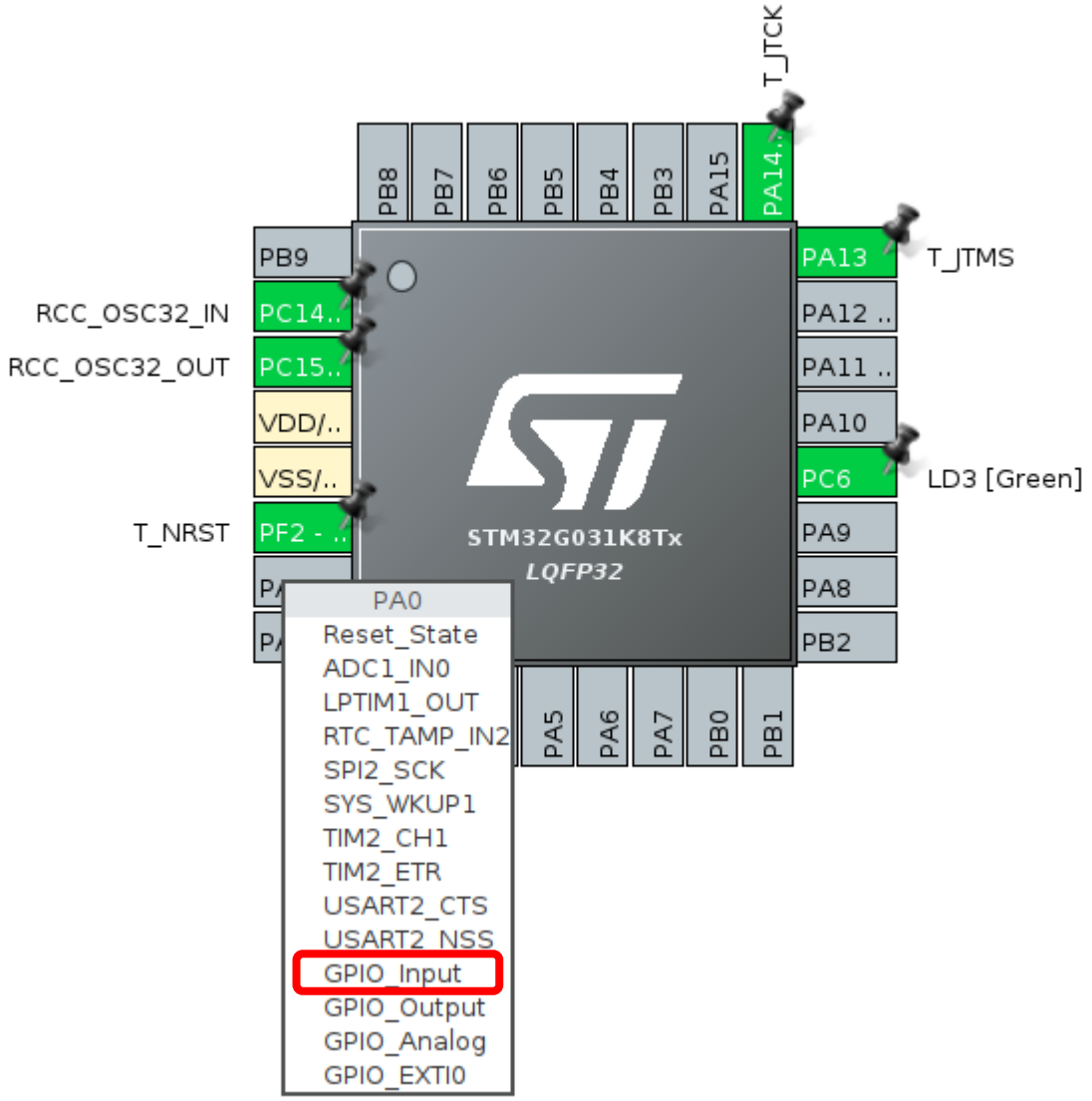


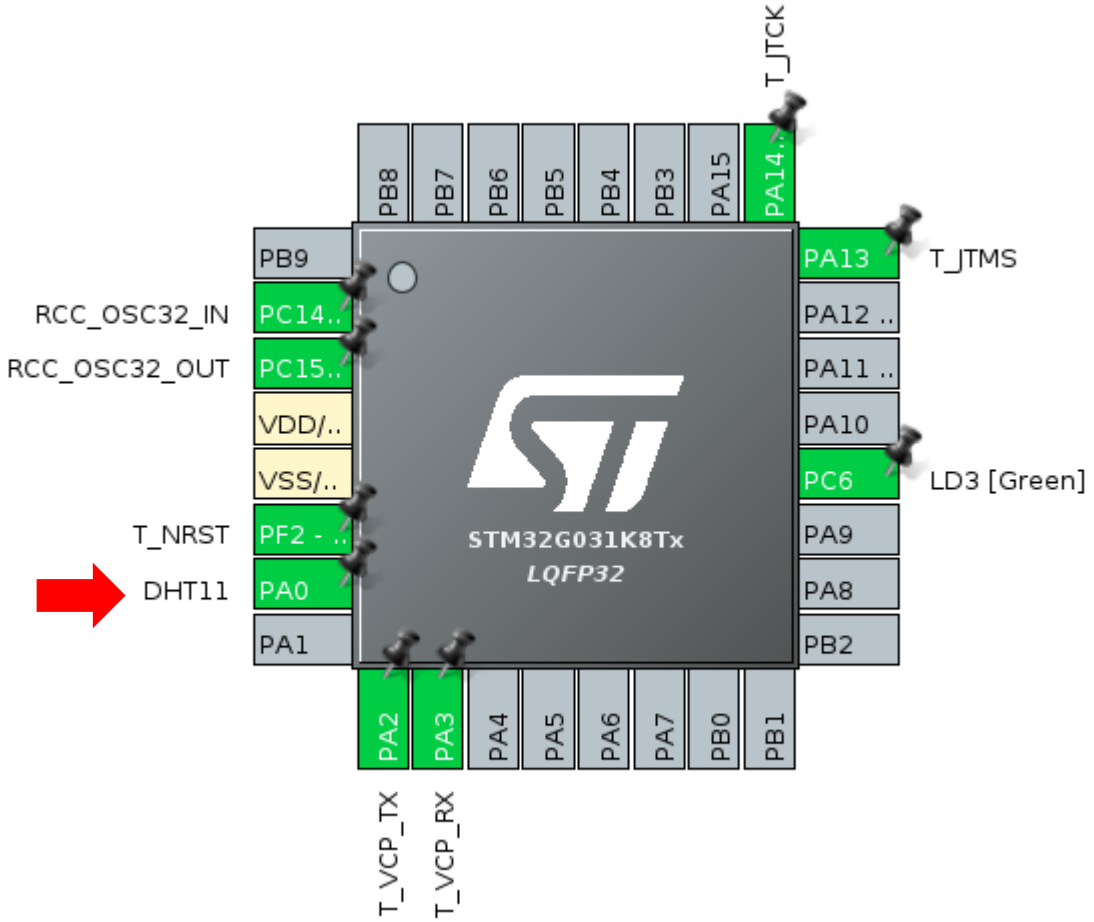
Configure the UART



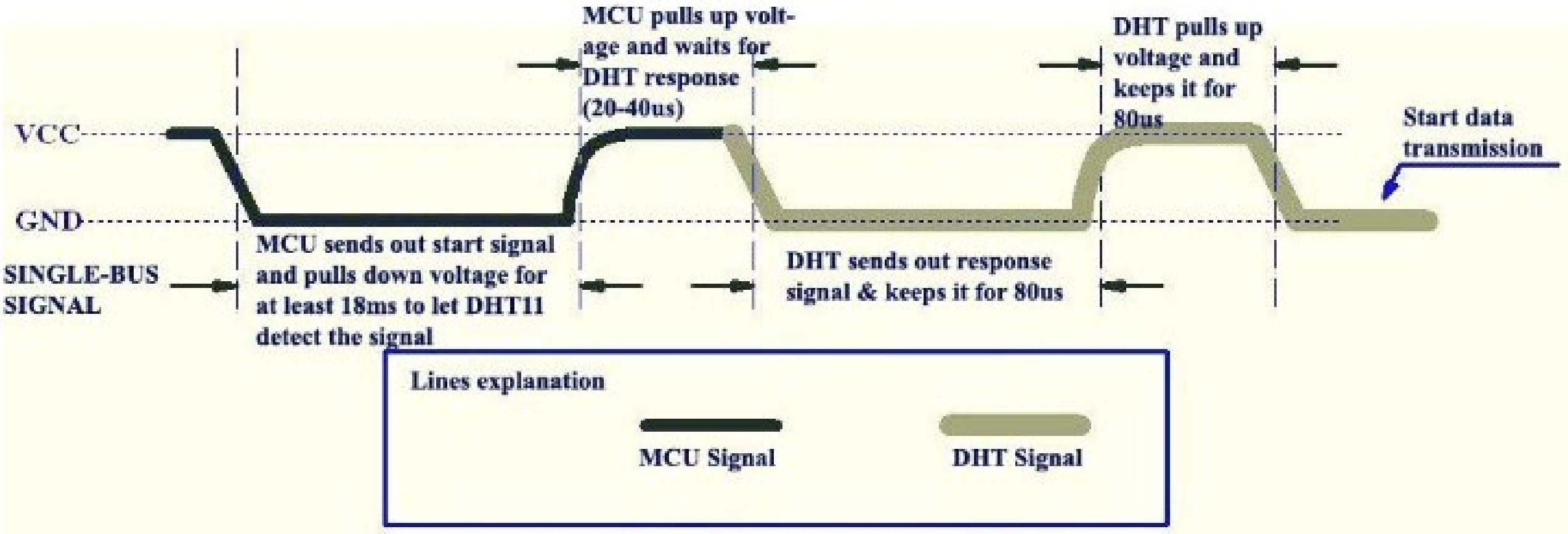


Configure the DHT11 GPIO Pin

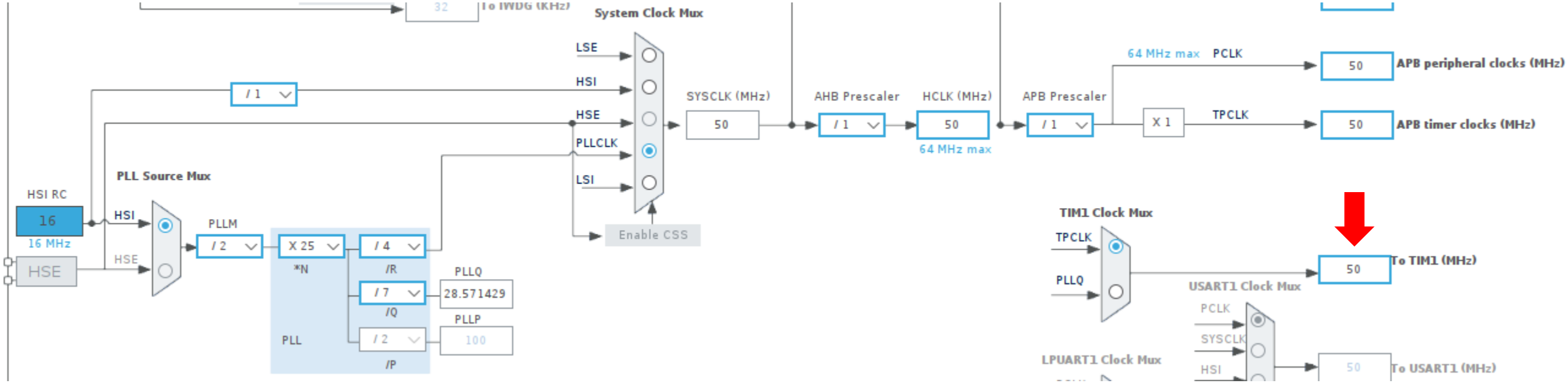




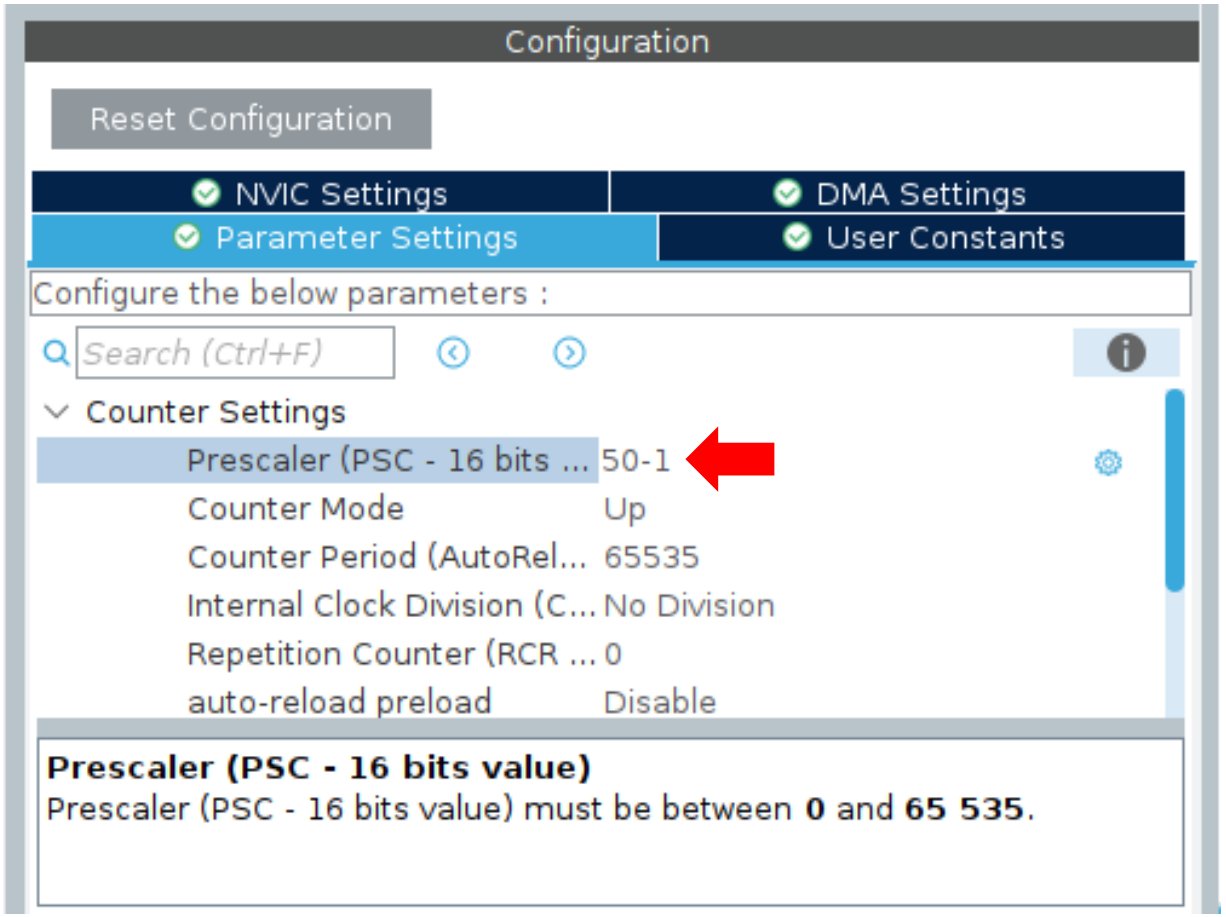
DHT11 Requires Microsecond Clock Resolution



Configure the G031K8 MCLK and TIM1 Timer Clocks

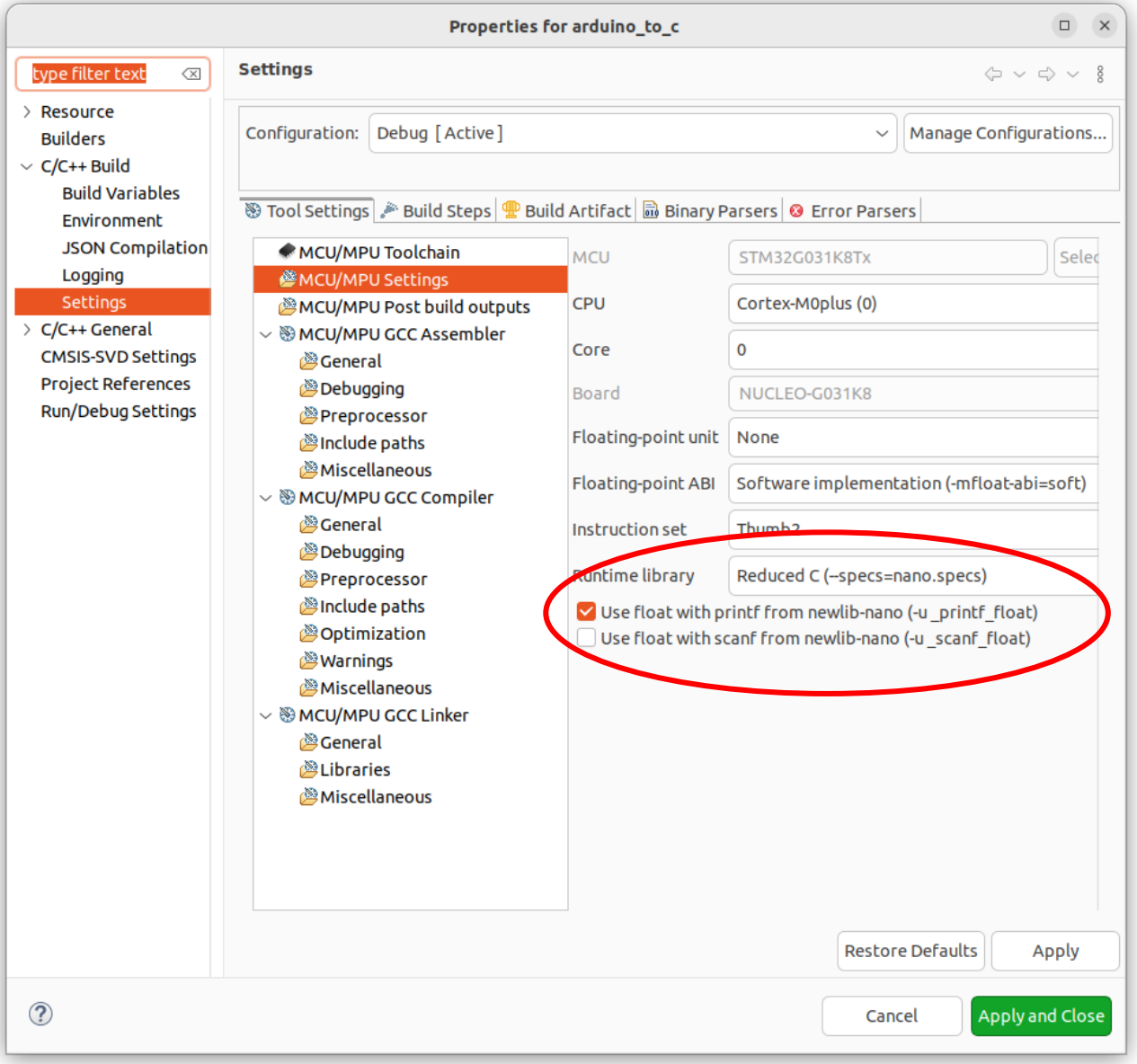


Configure the TIM1 as the Microsecond Delay Clock




```
57 /* Private user code ---------------------------------------------------------*/
58 /* USER CODE BEGIN 0 */
59 void delayus(uint16_t time)
60 {
61     __HAL_TIM_SET_COUNTER(&htim1,0);
62     while((__HAL_TIM_GET_COUNTER(&htim1))<time);
63 }
64 /* USER CODE END 0 */
```

Enable float with printf



DHT Class (In DHT.h)

```
63  class DHT {
64  public:
65    DHT(uint8_t pin, uint8_t type, uint8_t count = 6);
66    void begin(uint8_t usec = 55);
67    float readTemperature(bool S = false, bool force = false);
68    float convertCtoF(float);
69    float convertFtoC(float);
70    float computeHeatIndex(bool isFahrenheit = true);
71    float computeHeatIndex(float temperature, float percentHumidity,
72                           bool isFahrenheit = true);
73    float readHumidity(bool force = false);
74    bool read(bool force = false);
75
76  private:
77    uint8_t data[5];
78    uint8_t _pin, _type;
79  #ifdef __AVR
80    // Use direct GPIO access on an 8-bit AVR so keep track of the port and
81    // bitmask for the digital pin connected to the DHT.  Other platforms will use
82    // digitalRead.
83    uint8_t _bit, _port;
84  #endif
85    uint32_t _lastreadtime, _maxcycles;
86    bool _lastresult;
87    uint8_t pullTime; // Time (in usec) to pull up data line before reading
88
89    uint32_t expectPulse(bool level);
90  };
```

Translate pinMode and millis()

```
57  /*!
58   *  @brief  Setup sensor pins and set pull timings
59   *  @param  usec
60   *          Optionally pass pull-up time (in microseconds) before DHT reading
61   *starts. Default is 55 (see function declaration in DHT.h).
62   */
63  void DHT::begin(uint8_t usec) {
64    // set up the pins!
65    pinMode(_pin, INPUT_PULLUP);
66    // Using this value makes sure that millis() - lastreadtime will be
67    // >= MIN_INTERVAL right away. Note that this assignment wraps around,
68    // but so will the subtraction.
69    _lastreadtime = millis() - MIN_INTERVAL;   millis() = HAL_GetTick()
70    DEBUG_PRINT("DHT max clock cycles: ");
71    DEBUG_PRINTLN(_maxcycles, DEC);
72    pullTime = usec;
73  }
```

```
65  void set_DHTpin_input()
66  {
67  /*Configure GPIO pin : DHT11_Pin */
68  GPIO_InitTypeDef GPIO_InitStruct = {0};
69  GPIO_InitStruct.Pin = DHT11_Pin;
70  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
71  GPIO_InitStruct.Pull = GPIO_NOPULL;
72  HAL_GPIO_Init(DHT11_GPIO_Port, &GPIO_InitStruct);
73  }
74
75  void set_DHTpin_output()
76  {
77  /*Configure GPIO pin : DHT11_Pin */
78  GPIO_InitTypeDef GPIO_InitStruct = {0};
79  GPIO_InitStruct.Pin = DHT11_Pin;
80  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
81  GPIO_InitStruct.Pull = GPIO_NOPULL;
82  HAL_GPIO_Init(DHT11_GPIO_Port, &GPIO_InitStruct);
83  }
```

Translate DHT::read

```
224  /*!
225   *  @brief  Read value from sensor or return last one from less than two
226   *seconds.
227   *  @param  force
228   *          true if using force mode
229   *  @return float value
230   */
231  bool DHT::read(bool force) {
232    // Check if sensor was read less than two seconds ago and return early
233    // to use last reading.
234    uint32_t currenttime = millis();
235    if (!force && ((currenttime - _lastreadtime) < MIN_INTERVAL)) {
236      return _lastresult; // return last correct measurement
237    }
238    _lastreadtime = currenttime;
239
240    // Reset 40 bits of received data to zero.
241    data[0] = data[1] = data[2] = data[3] = data[4] = 0;
```

Translate DHT::read

```
247   // Send start signal.  See DHT datasheet for full signal diagram:
248   // Go into high impedence state to let pull-up raise data line level and
249   // start the reading process.
250   pinMode(_pin, INPUT_PULLUP);
251   delay(1);
252
253   // First set data line low for a period according to sensor type
254   pinMode(_pin, OUTPUT);
255   digitalWrite(_pin, LOW);
256   switch (_type) {
257   case DHT22:
258   case DHT21:
259     delayMicroseconds(1100); // data sheet says "at least 1ms"
260     break;
261   case DHT11:
262   default:
263     delay(20); // data sheet says at least 18ms, 20ms just to be safe
264     break;
265   }
266
267   uint32_t cycles[80];
268   {
269     // End the start signal by setting data line high for 40 microseconds.
270     pinMode(_pin, INPUT_PULLUP);
271
272     // Delay a moment to let sensor pull data line low.
273     delayMicroseconds(pullTime);
```

```
85  void DHT11_Start(void)
86  {
87      set_DHTpin_input();
88      HAL_Delay(1);
89      set_DHTpin_output();
90      HAL_GPIO_WritePin(DHT11_GPIO_Port,DHT11_Pin,GPIO_PIN_RESET);
91      HAL_Delay(20);
92      HAL_GPIO_WritePin(DHT11_GPIO_Port,DHT11_Pin,GPIO_PIN_SET);
93      set_DHTpin_input();
94      delayus(55);
95  }
```

Translate DHT::read

```
275     // Now start reading the data line to get the value from the DHT sensor.
276
277     // Turn off interrupts temporarily because the next sections
278     // are timing critical and we don't want any interruptions.
279     InterruptLock lock;
280
281     // First expect a low signal for ~80 microseconds followed by a high signal
282     // for ~80 microseconds again.
283     if (expectPulse(LOW) == TIMEOUT) {
284       DEBUG_PRINTLN(F("DHT timeout waiting for start signal low pulse."));
285       _lastresult = false;
286       return _lastresult;
287     }
288     if (expectPulse(HIGH) == TIMEOUT) {
289       DEBUG_PRINTLN(F("DHT timeout waiting for start signal high pulse."));
290       _lastresult = false;
291       return _lastresult;
292     }
```

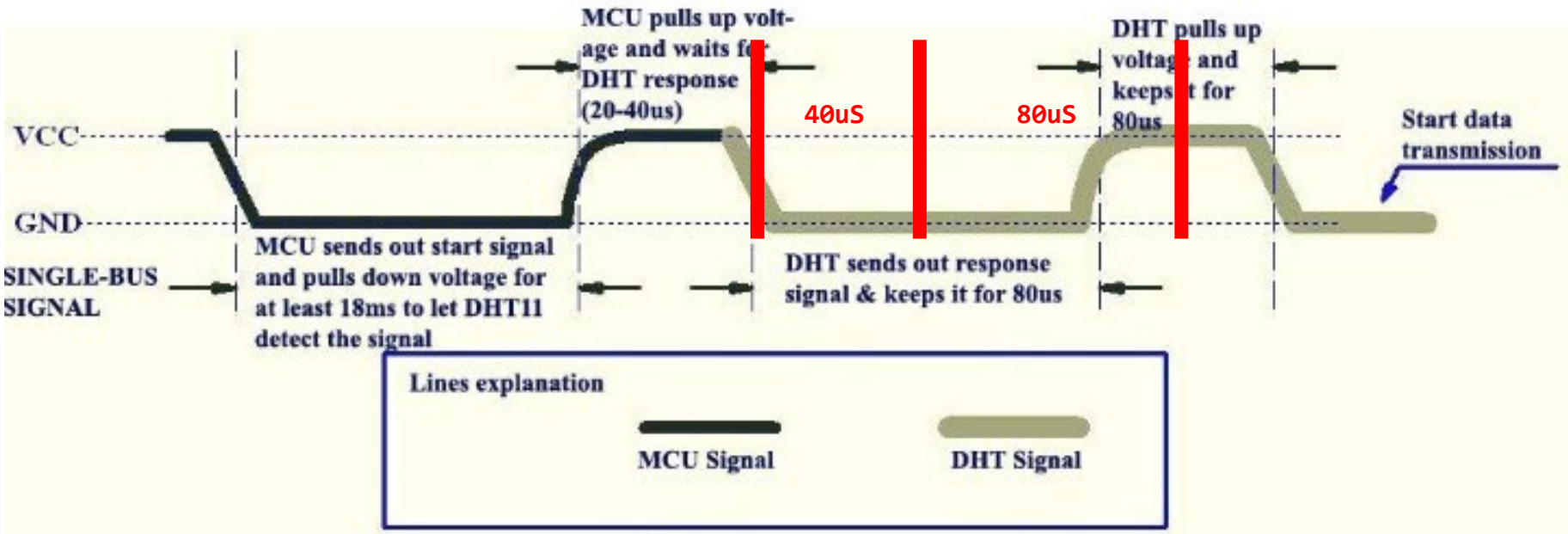
```
97  uint8_t DHT11_Start_Response(void)
98  {
99      uint8_t sensor_detected = 0;
100     delayus(40);
101     if(!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)))
102     {
103         delayus(80);
104         if((HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)))
105         {
106             sensor_detected = 1;
107         }
108     }
109     while ((HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
110     return sensor_detected;
111 }
```

Translate DHT::read

```
97  uint8_t DHT11_Start_Response(void)
98  {
99      uint8_t sensor_detected = 0;
100     delayus(40);
101     if(!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)))
102     {
103         delayus(80);
104         if((HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)))
105         {
106             sensor_detected = 1;
107         }
108     }
109     while ((HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
110     return sensor_detected;
111 }
```

40uS

80uS



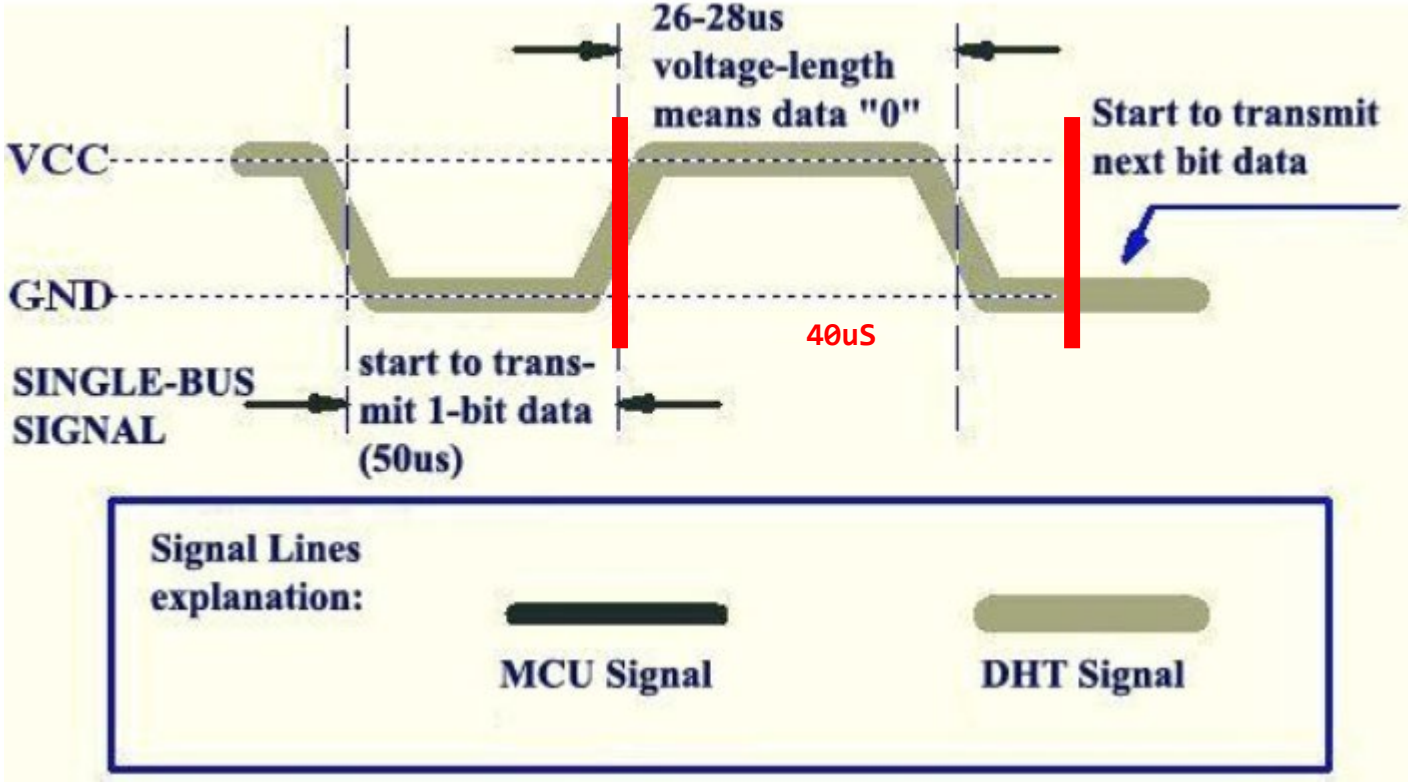
Translate DHT::read

```
294     // Now read the 40 bits sent by the sensor.  Each bit is sent as a 50
295     // microsecond low pulse followed by a variable length high pulse.  If the
296     // high pulse is ~28 microseconds then it's a 0 and if it's ~70 microseconds
297     // then it's a 1.  We measure the cycle count of the initial 50us low pulse
298     // and use that to compare to the cycle count of the high pulse to determine
299     // if the bit is a 0 (high state cycle count < low state cycle count), or a
300     // 1 (high state cycle count > low state cycle count). Note that for speed
301     // all the pulses are read into a array and then examined in a later step.
302     for (int i = 0; i < 80; i += 2) {
303       cycles[i] = expectPulse(LOW);
304       cycles[i + 1] = expectPulse(HIGH);
305     }
306   } // Timing critical code is now complete.
307
308   // Inspect pulses and determine which ones are 0 (high state cycle count < low
309   // state cycle count), or 1 (high state cycle count > low state cycle count).
310   for (int i = 0; i < 40; ++i) {
311     uint32_t lowCycles = cycles[2 * i];
312     uint32_t highCycles = cycles[2 * i + 1];
313     if ((lowCycles == TIMEOUT) || (highCycles == TIMEOUT)) {
314       DEBUG_PRINTLN(F("DHT timeout waiting for pulse."));
315       _lastresult = false;
316       return _lastresult;
317     }
318     data[i / 8] <<= 1;
319     // Now compare the low and high cycle times to see if the bit is a 0 or 1.
320     if (highCycles > lowCycles) {
321       // High cycles are greater than 50us low cycle count, must be a 1.
322       data[i / 8] |= 1;
323     }
324     // Else high cycles are less than (or equal to, a weird case) the 50us low
325     // cycle count so this must be a zero.  Nothing needs to be changed in the
326     // stored data.
327   }
```

```
113 uint8_t DHT11_Read(void)
114 {
115     uint8_t bite,bit;
116     for(bit=0;bit<8;bit++)
117     {
118         while (!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
119         delayus(40);
120         if(!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)))
121         {
122             bite&= ~(1<<(7-bit)); // bit=0
123         }
124         else
125         {
126         bite|= (1<<(7-bit)); // bit=1
127         }
128         while ((HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
129     }
130     return bite;
131 }
```

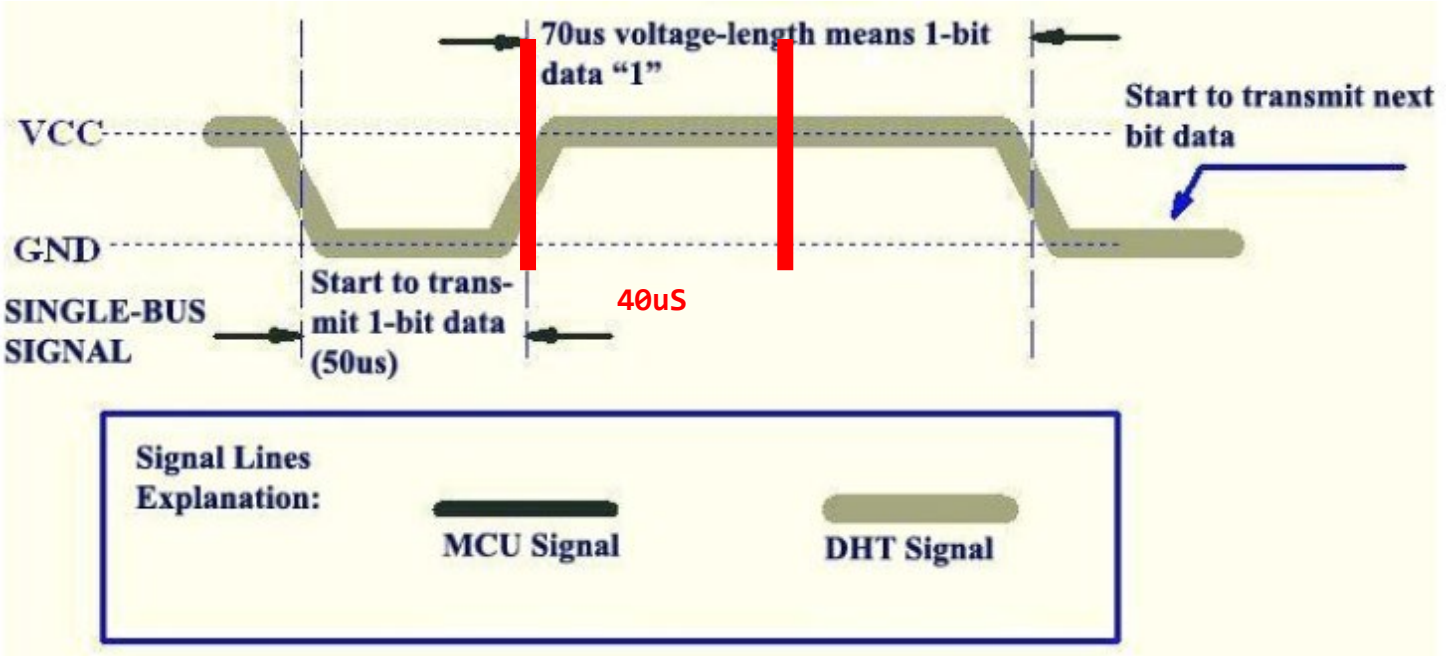
Translate DHT::read

```
113 uint8_t DHT11_Read(void)
114 {
115     uint8_t bite,bit;
116     for(bit=0;bit<8;bit++)
117     {
118         while (!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
119    40uS delayus(40);
120         if(!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)))
121         {
122             bite&= ~(1<<(7-bit)); // bit=0
123         }
124         else
125         {
126         bite|= (1<<(7-bit)); // bit=1
127         }
128         while ((HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
129     }
130     return bite;
131 }
```



Translate DHT::read

```
113 uint8_t DHT11_Read(void)
114 {
115     uint8_t bite,bit;
116     for(bit=0;bit<8;bit++)
117     {
118         while (!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
119 40uS    delayus(40);
120         if(!(HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)))
121         {
122             bite&= ~(1<<(7-bit)); // bit=0
123         }
124         else
125         {
126         bite|= (1<<(7-bit)); // bit=1
127         }
128         while ((HAL_GPIO_ReadPin(DHT11_GPIO_Port,DHT11_Pin)));
129     }
130     return bite;
131 }
```



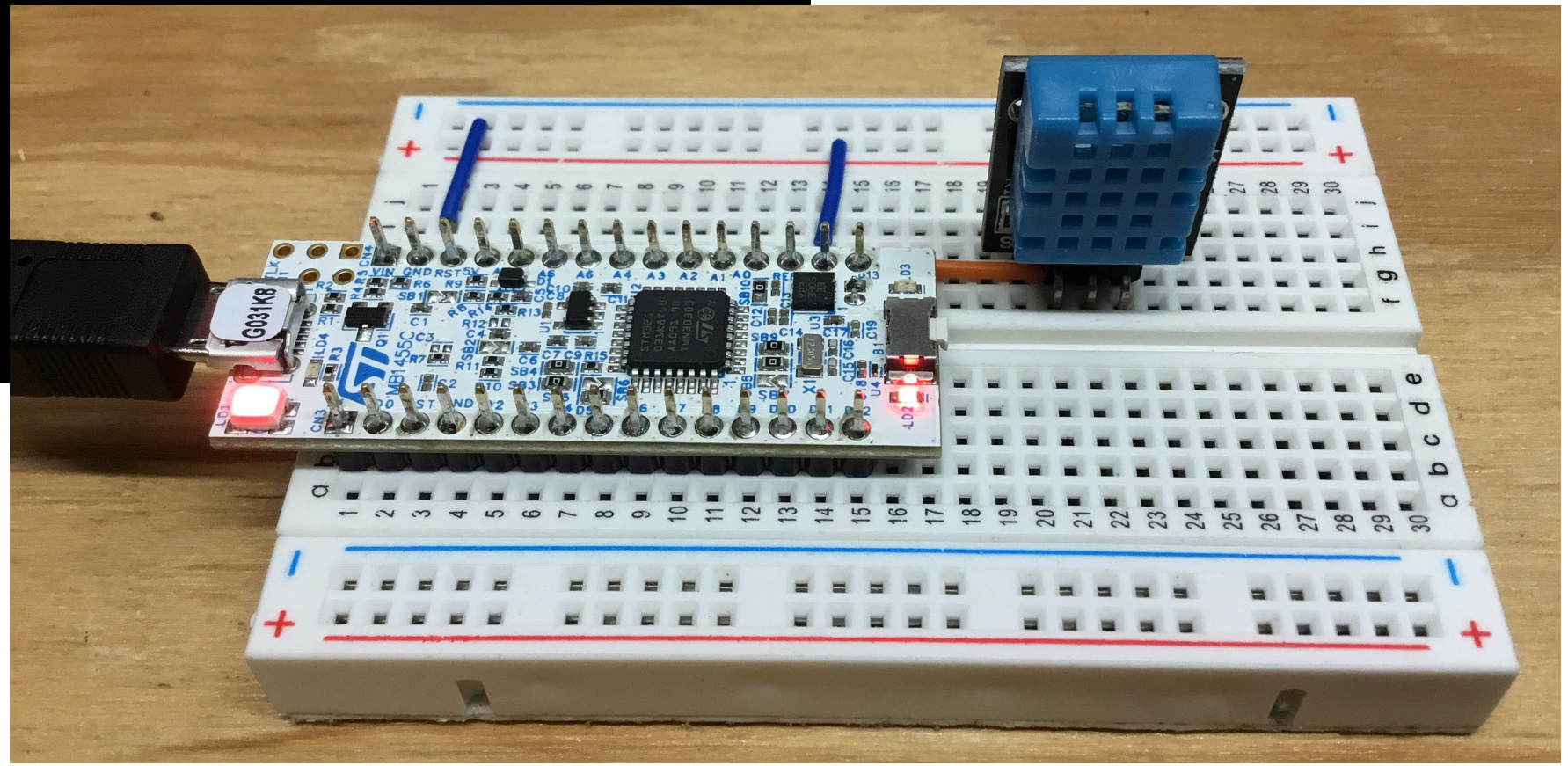
Run It!

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

+ Open an additional monitor

Monitor Mode: Serial | View Mode: Text | Port: /dev/ttyACM0 - STMicroelectronics | Baud rate: 115200

```
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
Temperature = 26.00 C
```



Thank you for attending!!!

MORE TO COME..

Please consider the resources below:

- Today's Download Package

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets