온도 센서 DS18B20 정보. Arduino 및 디지털 온도 센서 DS18B20 Dallas 18b20 배선도

이 센서는 1-Wire 프로토콜만 사용합니다. 이는 하나의 제어 신호만 사용하여 버스에서 통신하는 연결을 형성합니다. 버스는 풀업 저항을 통해 전원 공급 장치에 연결되어야 합니다.

사양 DS18B20

매개변수	의미
IC 출력 유형	디지털
감지 정확도 범위	± 0.5°C
온도 감지 범위	-55°C ~ +125°C
공급 전류	1mA
공급 전압 범위	3V ~ 5.5V
해상도(비트)	9...12
센서 케이스 스타일	TO-92
아니요. 핀의	3
기본 번호	18
작동 온도 최대	85°C
작동 온도 최소	-10°C
작동 온도 범위	-10°C ~ +85°C
출력 전류	4mA

악기

문

창문

타일

바닥

구성

건설

마지막 메모

문

인생에서 운이 좋지 않은 경우해야 할 일 : 행운을 불러오고 문제를 제거합니다.

문

출력 유형	디지털
패키지/케이스	TO-92
해결	9...12
센서/트랜스듀서 유형	온도
최대 공급 전압	5.5V
공급 전압 최소	3V
종료 유형	구멍
작동 온도, °C	0...+55
작동 상대 습도, %	...55
생산	달라스/맥심
작동 보증 기간	구매일로부터 12개월
무게, g	10

DS1820, DS18S20, DS18B20- 단일 와이어 1-Wire 인터페이스가 있는 DALLAS-MAXIM의 인기 있는 디지털 온도 센서. 아마추어 무선 문헌에 나타난 이러한 디지털 열 센서의 표시가 모호하고 회로가 많기 때문에 몇 가지 설명이 필요하다고 생각합니다.
칩 **DS1820 단종**마이크로 회로는 교체를 위해 권장됩니다. **DS18S20**. 그러나 마이크로 회로는 **TO-92 패키지의 DS18S20에는 "DS1820"이라는 레이블이 지정되어 있습니다(문자 S 제외).**. 새로운 칩 **이전 DS1820과 호환되는 DS18S20 소프트웨어**제조업체에 따르면 대부분의 경우 이전 DS1820을 직접 교체할 수 있습니다. 아마도 제조업체는 문자 S 없이 표시하여 이 호환성을 표시하기를 원했을 것입니다. 프로그램이 데이터 시트의 알고리즘을 사용하는 경우 새 DS18S20과 이전 DS1820의 소프트웨어 호환성이 보장됩니다.
표에서 볼 수 있듯이 새로운 DS18S20 칩은 표준 TO-92 패키지로 제작된 반면, 구형 DS1820은 패키지가 길쭉하게 만들어졌습니다. 이를 기반으로 판매자가 오래된 마이크로 회로로 귀하를 "밀지"않도록 할 수도 있습니다.
DS18B20 칩에는 항상 해당 마킹 "DS18B20"이 있으며 **프로그램 코드를 변경하지 않고 DS1820/DS18S20으로 또는 그 반대로 변경할 수 없습니다.**.

Maxim 디지털 온도 센서 사양

센서 유형	DS1820	DS18S20	DS18B20

팬케이크 주간: 그 단계

문

조디악의 가장 아름다운 징후의 등급

마킹	DS1820	DS1820	DS18B20
액자	PR-35 (내선 TO-92)	TO-92	TO-92
비트 깊이	9비트	9비트	9...12비트
전환 시간	200ms(일반) 500nS(최대)	750nS(최대)	750nS(최대)
측정 정확도 ±0.5% 온도 범위에서	0 ….+70°C	-10 ….+85°C	-10 ….+85°C
전원 전압 측정 정확도 ±0.5%	4.3-5.5V	3.0-5.5V	3.0-5.5V
설명			

마이크로 컨트롤러를 연구하는 과정에서 조만간 그러한 기상 매개 변수를 측정하는 것이 필요하게됩니다. **환경**그녀의 온도처럼. 전자 부품에 대한 현대적인 글로벌 시장은 광범위한 온도 센서를 제공합니다. 이들 사이의 주요 차이점은 측정된 온도, 공급 전압, 적용 영역, **전체 치수**, 온도 변환 방법, 사용자 제어 시스템과의 상호 작용을 위한 인터페이스. 역사적으로 가장 인기 있는 온도 센서 중 하나가 센서라는 것은 역사적으로 일어났습니다. **DS18B20**달라스반도체 다음 이야기는 그에 관한 것입니다.

DS18B20– **디지털 센서**프로그래밍 가능한 변환 분해능이 있는 온도.

고유 한 특징:

1) 1-Wire 인터페이스 데이터 버스를 사용하여 제어 시스템과 상호 작용
2) 내부 ROM 메모리에 있는 고유한 64비트 직렬 식별 코드가 있고 특정 센서를 지정해야 하는 다중 지점 시스템용입니다.
3) 공급 전압은 3-5.5V이므로 5볼트 시스템뿐만 아니라 3.3(대부분의 마이크로컨트롤러)에서도 사용할 수 있습니다.
4) 측정된 온도의 범위는 -55…+125 o C입니다.
5) ± 0.5 ° C의 정확도, 이것은 -10 ... + 85 ° C 범위에서만 해당됩니다.
6) 변환 해상도는 사용자가 결정하며 9…12비트입니다.

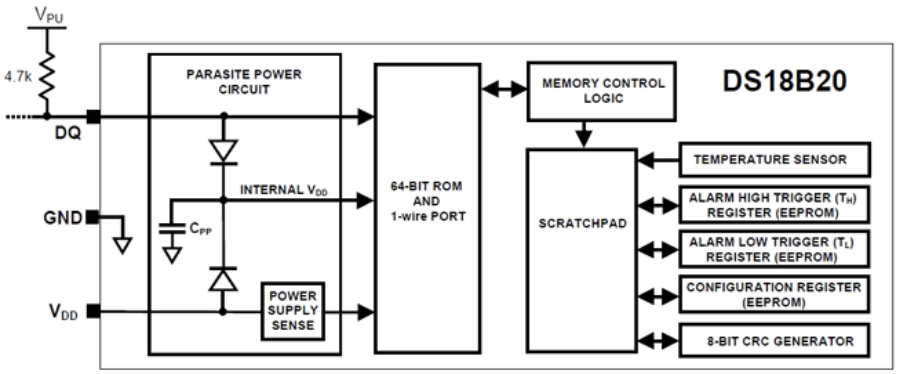
7) 온도 조절식 작동 논리를 사용하는 시스템에 대한 경보 신호 생성과 함께 상한 및 하한 임계값의 트리거 내부 레지스터가 있습니다.
8) 이 센서는 소프트웨어와 호환됩니다. **DS1822**산업용 자동 온도 조절 장치, 산업용 시스템, 소비자 전자 제품 및 기타 온도에 민감한 시스템에 널리 사용됩니다.

장치의 작동 원리 및 설명:

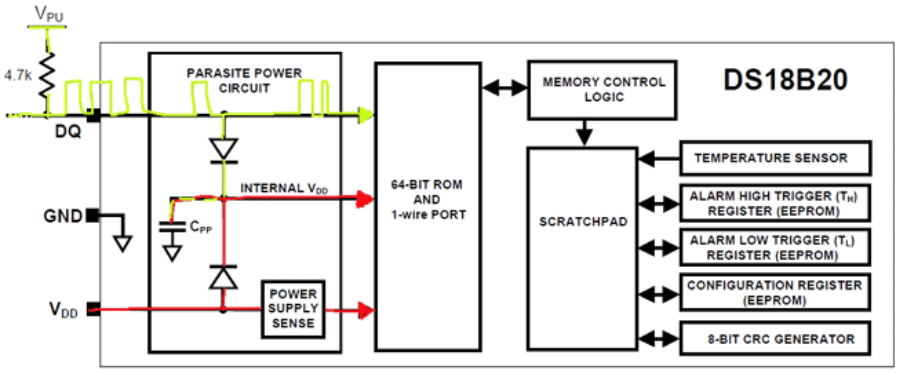
내 기사에서는 TO-92 패키지로 만든 센서로 작업하는 예를 설명합니다.

다음과 같이 보입니다.

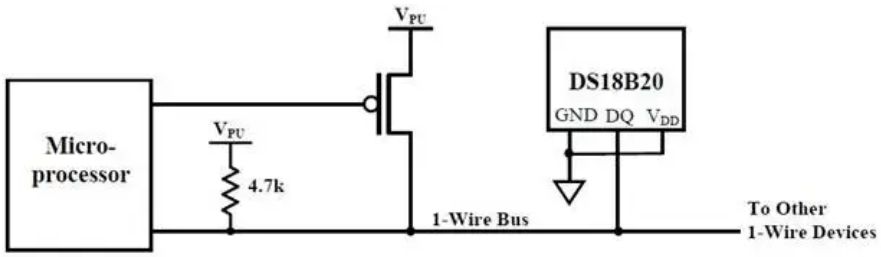
이 장치 내부는 매우 간단하게 배열되어 있습니다. 직접 살펴보세요.



이 블록다이어그램을 자세히 살펴보겠습니다.

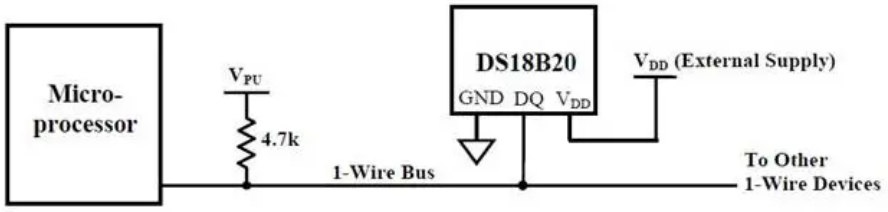


그러나 이러한 방식으로 전원을 공급하면 센서의 시간 매개변수에 몇 가지 제한이 따릅니다. 데이터 라인을 잠시 유지하면 커패시터가 방전되어 INTERNAL Vdd 라인의 전원이 차단되고 이에 따라 센서 전체가 비활성화됩니다. 따라서 사용하지 않는 시간 동안 DQ 라인을 높게 유지해야 합니다. 한 가지 중요한 언급이 있어야 합니다. 온도를 변환하고 Scratchpad에서 EEPROM으로 데이터를 복사할 때 (레지스터 중 하나로) INTERNAL Vdd 라인에 의해 소비되는 전류는 내부 커패시터가 견딜 수 없는 1.5mA에 도달할 수 있으며 풀 시 큰 전압 강하가 발생합니다. -업 저항, 일반적으로 장치의 작동에 허용할 수 없는 영향을 미칩니다. 이렇게하려면 다음 구성표에 따라 구현되는 강력한 풀업 구성표에서 DQ 라인을 구성해야 합니다.



명령을 내린 후 **전환하다티** 또는 **복사스크래치 패드** 센서 개발자에 따르면 최대 10μs(최대)의 MOSFET 트랜지스터로 DQ 라인의 강력한 풀업을 켜고 변환 시간(Tconv) 또는 데이터 전송 시간(Twr)을 기다려야 합니다. = 10ms), 이때 어떠한 조치도 취하지 않을 때 DQ 라인에 강력한 풀업이 켜져 있으면 안 됩니다!

여기에서는 모든 것이 간단하고 MOSFET조차도 전혀 필요하지 않기 때문에 표준 전력에 대해서는 거의 말할 필요가 없습니다.



"64-BIT ROM AND 1-Wire PORT" 서브시스템에는 비휘발성 ROM 메모리에 있는 고유한 64비트 직렬 식별 코드가 포함되어 있으며 이 노드에는 1-Wire 제어 시스템과의 상호작용을 위한 인터페이스도 포함되어 있습니다. "메모리 제어 로직" 서브시스템은 1-Wire 인터페이스 서브시스템과 Scratchpad 유형 메모리 사이에서 데이터 전송을 수행하며, 이 메모리는 차례로 온도 센서 레지스터, 상한 및 하한 경보 임계값 설정을 위한 레지스터, 구성, 레지스터 및 생성기 레지스터 8비트 체크섬을 사용하여 잘못된 데이터로부터 시스템을 보호합니다.

전원이 켜지면 센서는 기본적으로 12비트 변환 해상도로 설정되고 즉시 저전력 모드로 들어갑니다. 변환을 시작하려면 마스터가 명령을 보내야 합니다. **전환하다티 .** 온도를 디지털 코드로 변환한 후 이 코드는 스크래치패드 메모리에 2바이트 워드로 저장되고 센서는 절전 모드로 다시 전환됩니다.

온도 변환.

이제 센서에서 온도가 어떻게 변환되는지 알아봅시다. 실제로 ADC는 온도 센서 자체 내부에 있으며 온도 레지스터에 있는 출력 데이터는 Scratchpad 메모리로 전송됩니다. 온도 데이터의 형식은 다음과 같습니다.

S 플래그는 숫자의 부호를 나타내는 데 사용되는 부호 플래그입니다(S=0은 비트 10-0에 포함된 숫자가 양수이고 동일한 비트에 포함

된 숫자가 음수인 경우 S=1입니다. **이 경우**온도는 2의 보수 코드(2의 보수 코드)로 표시됩니다.

12비트 변환 해상도로 설정하면 모든 12비트(비트 11비트 0)가 활성화되고 유효한 데이터가 포함됩니다. 11비트로 설정하면 비트 0의 내용이 무시되어야 하고, 10비트로 설정되면 비트 0과 1이 고려되지 않아야 하는 식입니다.

알람 신호는 온도 조절기의 기능입니다.

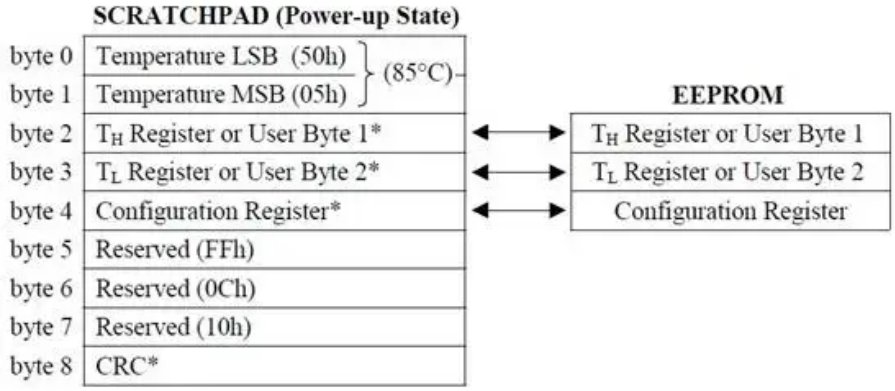
이를 위해 2개의 8비트 레지스터 Th 및 TI이 제공됩니다. Th는 각각 상위 온도 임계값의 값을 포함하고 TI은 하위 임계값을 포함합니다. 온도가 Th보다 높거나 TI보다 낮으면 알람 플래그가 설정됩니다. 이 알람 플래그는 명령을 실행하여 마스터가 감지합니다. **알람 검색** DQ 라인에. 알람 플래그는 각 온도 변환 작업 후에 업데이트됩니다. 덧붙여서, Th 또는 TI 레지스터와 비교하여 온도 레지스터의 비트 11 ~ 4만 사용됩니다. 즉, 온도 조절기 기능은 정수 온도에서만 작동합니다. 레지스터는 물리적으로 EEPROM 메모리이므로 전원이 꺼질 때 값을 유지합니다. 레지스터 자체는 온도 레지스터와 유사하지만 8비트일 뿐이며 S 플래그는 이전 경우와 정확히 동일한 의미를 갖습니다.

앞서 언급했듯이 이 코드는 다지점 온도 측정 시스템에서 라인의 각 장치를 식별하는 데 필요합니다.

이 메모리의 형식은 다음과 같습니다.

하위 8비트는 제품군 지정을 위해 예약되어 있으며 값 0x28을 포함하고 다음 48비트는 장치의 고유한 일련 번호를 포함합니다. 최상위 바이트는 ROM 메모리의 하위 56비트에 대해 계산된 CRC 체크섬 값을 포함합니다.

기억의 조직.



트랜스미터의 메모리는 Scratchpad 메모리 공간과 구성 데이터와 상한 및 하한 경보 레지스터 값을 저장하기 위한 EEPROM으로 구성됩니다.

전원이 꺼지면 데이터 바이트 2, 3, 4는 EEPROM에서 값을 유지합니다. 음, 켜면 값이 변경되지 않습니다. 바이트 0과 1은 변환된 온도 값을 포함하고 바이트 5, 6, 7은 내부 사용을 위해 예약되어 있으며 사용자의 필요에 따라 액세스할 수 없습니다.

8번째 바이트는 바이트 0에서 7에 대해 내장된 CRC 생성 로직에 의해 생성된 값을 포함하며, 이는 결국 잘못된 온도 판독 가능성을 최소화합니다.

온도 조절기 기능을 사용하지 않는 경우 Th 및 Tl 레지스터를 범용 메모리로 사용할 수 있습니다. 모든 정보를 저장할 수 있습니다.

데이터는 명령어를 사용하여 바이트 2의 최하위 비트부터 시작하여 바이트 2, 3, 4에 기록됩니다. **스크래치 패드 쓰기**. 기록된 데이터의 무결성을 확인하기 위해 읽을 수 있습니다. 이를 위해 센서에 명령을 보내야 합니다. **스크래치패드 읽기**, 그 후에 마스터는 바이트 0의 최하위 비트부터 데이터를 수신해야 합니다.

EEPROM 메모리의 구성 레지스터뿐만 아니라 온도 조절기의 상위, 하위 레지스터 데이터를 저장하려면 마스터 장치가 센서에 명령을 보내야 합니다. **스크래치 패드 복사.**

앞서 언급했듯이 EEPROM에 이미 기록된 데이터는 전원이 꺼질 때 유지됩니다. 그러나 전원이 켜지면 해당 EEPROM 셀의 값이 해당 스크래치 패드 메모리 레지스터에 자동으로 로드됩니다. 편리하죠? :)

또한 EEPROM에 기록된 데이터는 언제든지 스크래치패드 메모리에 덮어쓸 수 있습니다. 이것은 예를 들어 작동 중에 구성을 변경한 다음 "일반 모드"로 전환해야 할 때 필요합니다. 스크래치 패드 메모리 레지스터의 내용이 변경되기 전의 작업 구성을 반환합니다. 실제로 이를 위해서는 마스터 장치가 센서에 명령을 보내야 합니다. **리콜 E2 .**

구성 레지스터에서 사용자는 R0과 R1의 2비트만 정의할 수 있습니다. 이 비트는 온도 변환 분해능을 결정하며 기본적으로 1로 설정되며, 이는 12비트 변환 분해능의 초기 설정입니다.

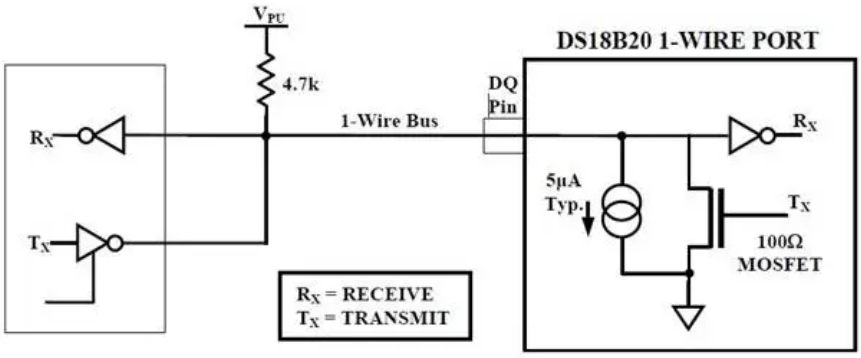
이러한 비트의 가능한 모든 구성과 해당 권한은 아래 표에 나와 있습니다. 변환 분해능이 클수록 변환 시간이 길어집니다. 예를 들어 12비트 분해능의 경우 변환 시간은 750ms(최대)입니다.

R1	R0	Resolution	Max Conversion Time	
0	0	9-bit	93.75 ms	(t_{CONV}/8)
0	1	10-bit	187.5 ms	(t_{CONV}/4)
1	0	11-bit	375 ms	(t_{CONV}/2)
1	1	12-bit	750 ms	(t_{CONV})

제어 시스템과의 상호 작용.

앞서 언급한 바와 같이 DS18B20은 1-Wire 인터페이스 데이터 버스를 사용하여 슬레이브 소자와 통신한다. 따라서 이를 연결하려면 제어 시스템이 출력을 개방 드레인 또는 Hi-Z 라인 상태로 제공해야 합니다.

센서 인터페이스의 내부 구성은 다음과 같습니다.



비활성 상태(유휴 상태)에서 DQ 라인은 "+" 전원 공급 장치에 대한 저항에 의해 풀업됩니다. 따라서 트랜잭션(데이터 전송) 간에 이 라인은 항상 이 상태로 유지되어야 합니다. 어떤 이유로든 거래가 일시 중단되어야 하는 경우, 이 전송이 더 재개되려면 DQ 라인이 높게 유지되어야 합니다. 트랜잭션을 중지하는 과정에서 DQ 라인을 1 µs부터 시작하여 원하는 만큼 높은 논리적 수준으로 유지할 수 있습니다. 그러나 데이터 버스가 480µs 이상 로우 상태로 유지되면 이 버스에 있는 모든 센서가 완전히 재설정됩니다.

교환 작업의 순서입니다.

신청할 때마다 **제어 시스템**센서에 대해 다음과 같은 일련의 작업을 준수해야 합니다.

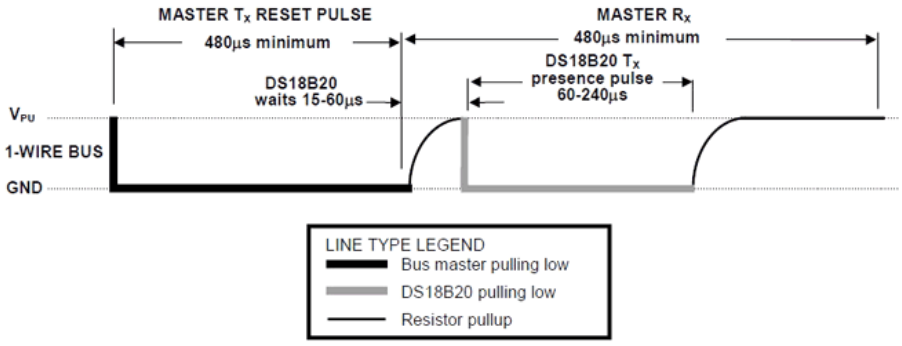
1) 초기화;
2) ROM 명령(필요한 데이터 교환이 뒤따름);
3) 센서 기능 명령(필요한 통신이 뒤따름).

센서에 접근할 때 단계가 없으면 센서가 응답하지 않습니다. 예외는 명령입니다 **검색ROM [에프0 시간]** 그리고 **경보검색 [ECH] ,** 실행 후 마스터는 제어 시퀀스의 첫 번째 단계로 돌아가야 합니다.

그래서. 모든 트랜잭션은 초기화로 시작됩니다. 이 작업은 마스터에 의해 리셋 펄스가 생성되고 슬레이브(이 경우 센서)는 마스터에 존재 펄스를 전송하여 센서가 연결되어 있고 사용할 준비가 되었음을 알립니다. 작업.

일반적으로 센서에 구현된 1-Wire 인터페이스 버스는 데이터 라인에서 여러 유형의 신호를 정의합니다. 리셋 펄스, 존재 펄스, 쓰기 0,

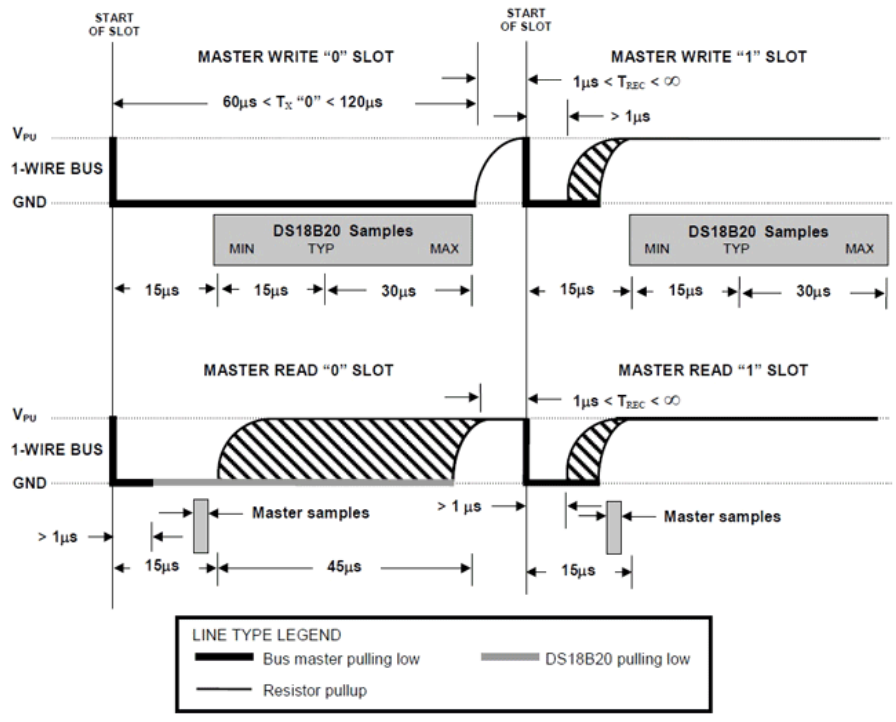
쓰기 1, 읽기 0, 읽기 1. 이러한 모든 작업은 마스터 장치에 의해 구현됩니다. , 존재 펄스를 제외하고. 센서에 의해서만 형성됩니다.



따라서 우선 마스터는 송신기 모드로 전환하고 DQ 라인을 최소 480µs 동안 0으로 설정합니다(굵은 검은색으로 강조 표시됨). 이것은 센서를 재설정합니다. 그런 다음 라인은 해제되어야 하고 마스터 장치는 수신기 모드로 들어가야 하며, 풀업 저항은 데이터 라인을 로직 하이 레벨(얇은 검정색으로 강조 표시됨)으로 설정합니다. 센서가 상승 에지를 감지한 후 센서는 15-60µs를 기다리고 하드웨어 인터페이스에 의해 데이터 라인을 0으로 재설정하고 60-240µs 동안 유지합니다. 이 시간이 지나면 센서가 라인을 해제하고 센서가 리셋 펄스를 감지한 후 최소 480µs 동안 논리 1 레벨로 설정됩니다.

이제 데이터 전송 프로세스가 수행되는 방식에 대해 이야기해 보겠습니다. 일반적으로 비트 전송. 요점은 다음과 같습니다. 일정 시간이 걸리고 이 시간 동안 마스터는 우리가 회선에 있는 것을 살펴봅니다. 1이라고 가정해 봅시다. 1을 적었다는 의미이고 0이면 0을 적었다는 의미입니다. 그러나 이것은 추상적인 설명일 뿐입니다. 사실, 이 전체 사건의 기간과 관련된 약간의 뉘앙스가 있습니다.

사진 보기:



모든 것은 마스터가 데이터 라인을 로직 로우 레벨로 낮추어야 하고 그 순간부터 1/0 쓰기/읽기 슬롯이 시작되어 60~120μs 동안 지속된다는 사실에서 시작됩니다. 쓰기/읽기 슬롯 사이에서 데이터 라인은 복구 시간(1μs) 이상의 시간 동안 1로 설정되어야 합니다. 쓰기 슬롯 0을 구성하려면 슬롯의 항상 데이터 라인을 0으로 유지해야 하지만 센서 1에 써야 하는 경우 먼저 데이터 라인을 0으로 재설정한 다음 적어도 1을 기다립니다. μs를 입력하고 1에서 라인을 해제하면 쓰기 슬롯 1(60-120μs)이 센서에 1을 씁니다(오른쪽 상단 그림 참조).

실제로 시작 후 15-60 μs 이내에 데이터 라인에서 1이 감지되면 1이 기록되고 60-240 μs 이내에 0이 감지되면 0이 기록됩니다.

데이터 판독은 마스터 장치와 함께 수행되며, 라인을 재설정할 때 최소 1μs를 기다리며 15μs 동안 라인에서 어떤 일이 발생하는지 살펴봅니다. 0이 남아 있으면 센서가 0을 전송하고 1, 그 다음 1이 전송되었습니다.

팀.

ROM 명령.

이러한 명령은 초기화 순서를 따라야 하며 적절한 센서 등을 찾기 위한 지침을 포함해야 합니다. 각 명령어의 용량은 8비트입니다. 적절한 명령을 실행한 후 센서에 기능 명령을 보낼 수 있습니다.

검색롬	시스템이 처음 연결되면 버스에 연결된 모든 장치를 인식해야 합니다. 그것이 이 명령의 목적입니다. 그러나 센서가 하나뿐이므로 이 명령을 사용하지 않습니다.
롬 읽기	이 명령은 버스에 센서가 하나만 있는 경우에만 사용됩니다. 이를 통해 마스터 장치는 find 명령을 사용하지 않고 64비트 ROM 메모리의 내용을 읽을 수 있습니다. 그리고 2개 이상의 연결된 센서와 함께 이 명령을 사용하려고 하면 모든 센서가 이 메모리의 내용을 전송하기 시작하여 바람직하지 않은 결과를 초래할 것입니다.
매치롬	이것은 ROM 일치 명령입니다. 마스터는 버스에 연결된 센서의 해당 ROM의 64비트를 해제하고 이미 수행할 작업(온도 측정 등)을 결정합니다. 이 때 버스의 다른 센서는 차례를 기다립니다.
스킵롬	이것은 ROM 건너뛰기 명령입니다. 버스에 있는 특정 센서의 주소를 고려하지 않고 한 번에 주소를 모두 입력합니다. 이 명령 후에, 예를 들어 온도 변환 명령을 실행하면 모든

	센서가 변환을 시작합니다. 그러나 이 명령을 호출한 후 메모리 읽기 명령을 실행하면 예측할 수 없는 결과가 발생합니다(모든 센서가 한 번에 데이터를 전송하기 때문). 즉, 연결된 센서 하나만 있으면 이러한 상황이 가능합니다.
알람 검색	이 명령은 알람 플래그가 설정된 버스에서 센서를 검색한다는 점을 제외하고 이 표의 첫 번째 명령과 동일합니다.

기능적 명령.

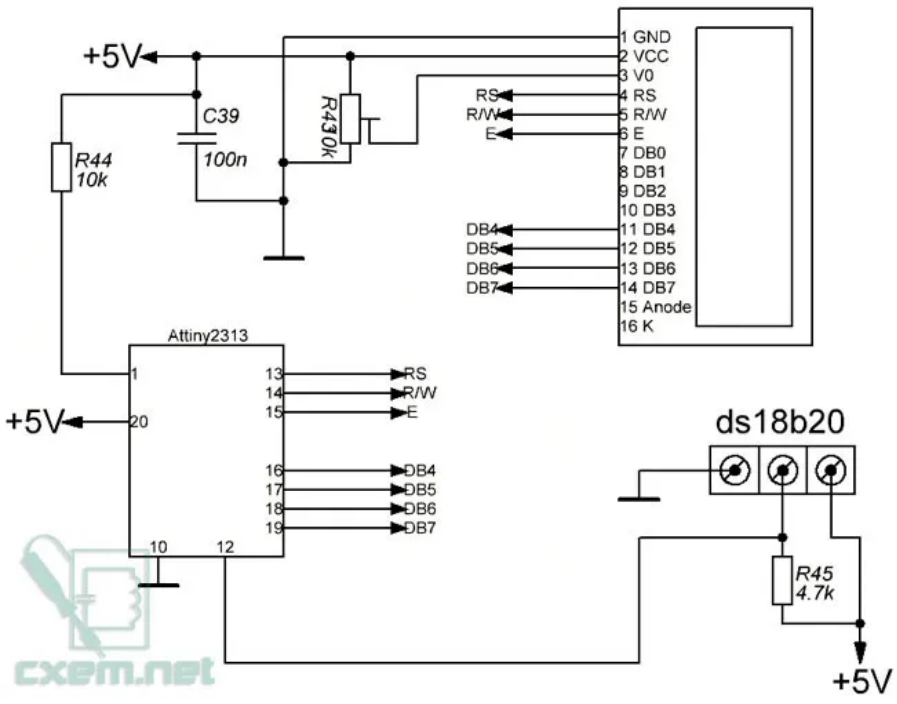
이 명령은 온도 변환 작업 시작, 메모리 복사 등과 같은 모든 프로세스의 기능적 작업을 수행합니다. 총 6개의 명령이 있으며 각 비트 길이는 8비트입니다.

변환 T	온도 변환을 시작합니다. 이 명령을 실행한 후 2바이트 데이터가 온도 레지스터에 입력됩니다.
스크래치패드 쓰기	두 번째 최하위 비트부터 시작하여 레지스터 2-4에 데이터를 씁니다. 3개의 레지스터로 데이터를 전송하는 동안 데이터가 손실될 수 있으므로 마스터가 센서를 재설정하지 않도록 주의해야 합니다.
스크래치패드 읽기	바이트 0의 하위 비트에서 시작하여 바이트 8(CRC)의 상위 비트로 끝나는 모든 스크래치패드 메모리 레지스터에 대한 데이터 전송 프로세스를 시작합니다.
스크래	이 명령어는 바이트 레지스터 2, 3, 4의 내용을 적절한 EEPROM 위치에 복사합니다.

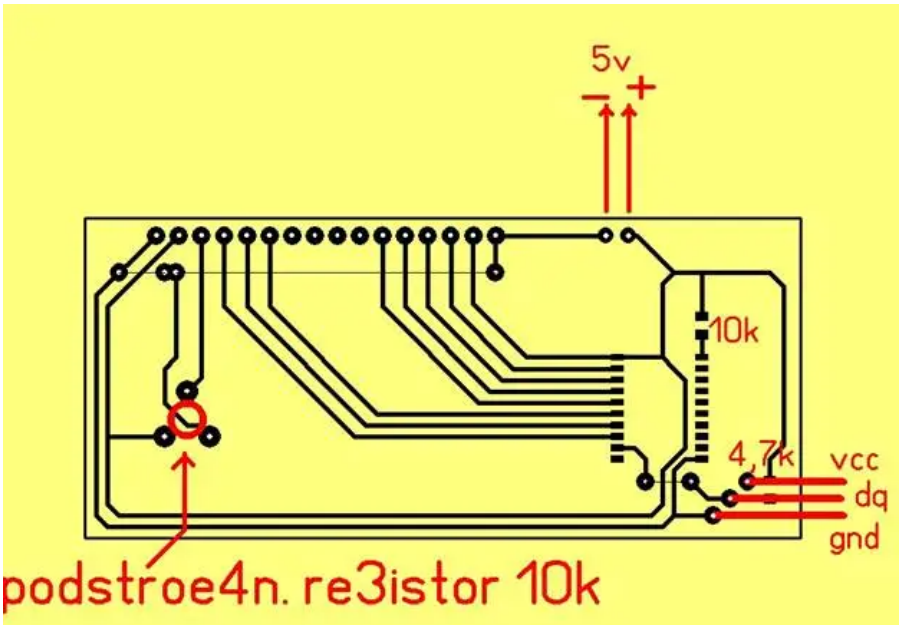
치 패 드 복 사	
리 콜 E2	이 명령은 EEPROM에서 스크래치 패드의 적절한 위치로 데이터를 복사합니다. 앞에서 언급한 것처럼 이 작업은 전원을 켜면 자동으로 발생합니다.
전 원 공 급 장 치 읽 기	

사실 이것이 DS18B20 온도 센서로 작업할 때의 지혜입니다. 자세한 내용은 데이터 시트()를 참조하십시오. 이제 이 모든 사업을 철로 구현해야 합니다.

장치의 개략도:



인쇄 회로 기판의 어셈블리 도면(품질에 대해 죄송합니다. 디버깅을 위해 작동하도록 했습니다):



보드를 올바르게 미러링하는 것을 잊지 마십시오.

이것은 브레드보드이기 때문에 이전 프로젝트에서 꺼냈으므로 위의 보드에서 내가 가지고 있는 것과 약간 다릅니다(내에서는 이제 불필요한 모든 것을 제거하고 위의 그림과 똑같이 되었습니다).

나에게 일어난 일은 다음과 같습니다.

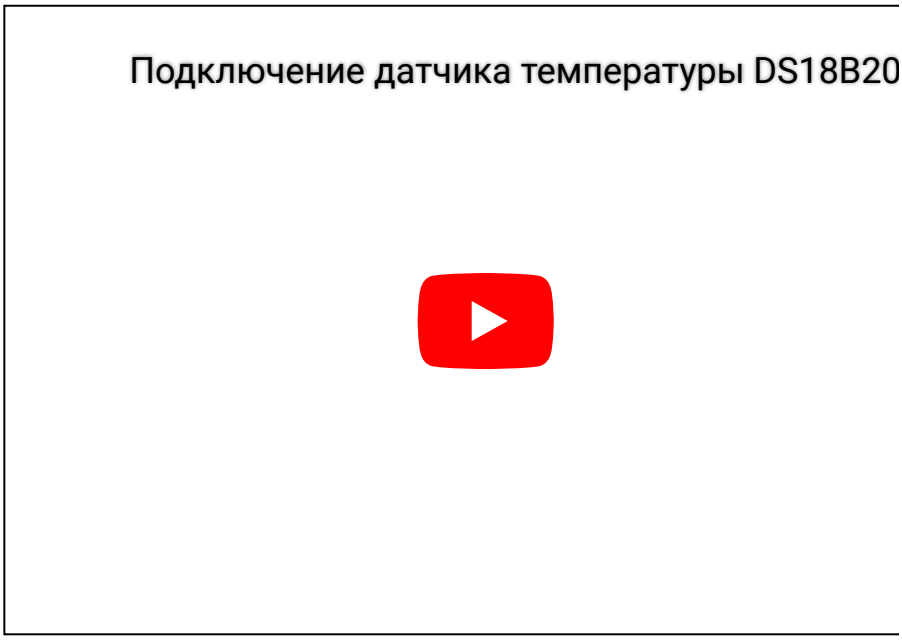
샌드위치가 되었어요.

프로그램의 소스 코드는 개발 환경에서 작성되었습니다. 나는 기성품 avr-gcc 컴파일러 라이브러리를 최대한 사용하려고 시도하지 않았지만 "손으로"말하는대로 모든 것을 작성했습니다. 제 목표는 C 기교를 시연하는 것이 아니라 초보자에게 센서 작업에 대한 일반적

인 아이디어를 제공할 수 있는 한 시간 안에 작성된 예시일 뿐입니다.
이 장치는 실내에서 사용하기 위한 것이므로 음의 온도 측정을 제공하지 않습니다.

소스 다운로드 및 **인쇄 회로 기판** LAY 당신은 아래에 할 수 있습니다

추가 질문이나 제안은 다음에서 환영합니다.
stalkerelectronics@mail.ru



Arduino의 온도 센서는 가장 일반적인 유형의 센서 중 하나입니다. Arduino 온도계 프로젝트 개발자가 사용할 수 있는 옵션이 많이 있습니다. **다른 옵션**, 작동 원리, 정확성, 디자인이 다릅니다.
DS18B20 디지털 센서는 물 또는 기타 액체의 온도를 측정하기 위해 방수 하우징에 자주 사용되는 가장 널리 사용되는 온도 센서 중 하나입니다. 이 기사에서는 러시아어로 된 ds18b20 센서에 대한 설명을 찾을 수 있으며 arduino에 연결하는 기능, 센서 작동 원리, 라이브러리 및 스케치에 대한 설명을 함께 고려할 것입니다.

DS18B20은 많은 유용한 기능을 갖춘 디지털 온도 센서입니다. 실제로 DS18B20은 측정 값을 저장하고, 설정된 한계를 초과하는 온도에 신호를 보내고(한계를 설정 및 변경할 수 있음), 측정 정확도, 컨트롤러와의 상호 작용 방식 등을 변경할 수 있는 전체 마이크로컨트롤러입니다. 이 모든 것이 매우 작은 패키지에 담겨 있으며 방수 버전으로도 제공됩니다.

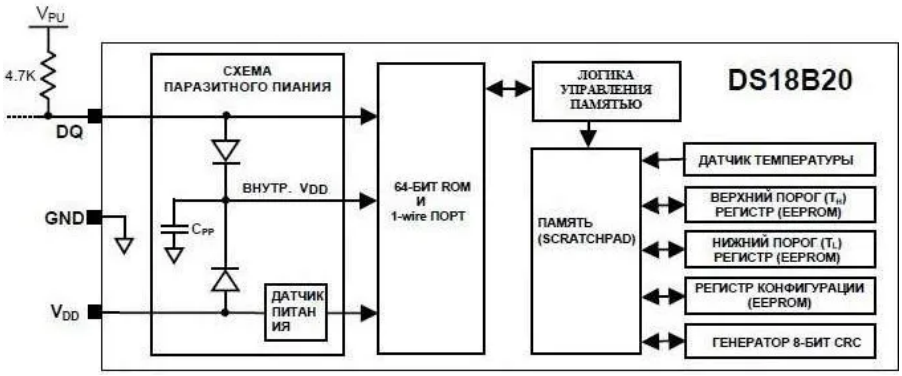
DS18B20 온도 센서에는 다양한 하우징 유형이 있습니다. 8핀 SO(150mils), 8핀 µSOP 및 3핀 TO-92의 세 가지 중에서 선택할 수 있습니다. 후자가 가장 일반적이며 특수 방수 케이스로 제작되어 수중에서도 안전하게 사용할 수 있습니다. 각 센서에는 3개의 핀이 있습니다. TO-92 케이스의 경우 검정색 - 접지, 빨간색 - 전원 및 흰색 /

노란색 / 파란색 - 신호와 같은 전선의 색상을 확인해야 합니다. 온라인 상점에서 기성품 DS18B20 모듈을 구입할 수 있습니다.

센서 구입처

당연히 DS18B20은 arduino가 있는 러시아 전문 온라인 상점에서도 판매되지만 Aliexpress에서 가장 저렴합니다. 다음은 예시에 대한 몇 가지 링크입니다.

센서 메모리는 작동 및 비휘발성의 두 가지 유형(SRAM 및 EEPROM)으로 구성됩니다. 후자는 구성 레지스터와 TH, TL 레지스터를 포함하며, 허용 온도 값 범위를 나타내는 데 사용되지 않는 경우 범용 레지스터로 사용할 수 있습니다.



DS18B20의 주요 임무는 온도를 결정하고 그 결과를 디지털 형식으로 변환하는 것입니다. 정밀도 비트 수(9, 10, 11 및 12)를 설정하여 필요한 분해능을 독립적으로 설정할 수 있습니다. 이 경우 분해능은 각각 0.5C, 0.25C, 0.125C 및 0.0625C와 같습니다.

수신된 온도 측정값은 센서의 SRAM에 저장됩니다. 바이트 1과 2는 수신된 온도 값을 저장하고, 3과 4는 측정 한계를 저장하고, 5와 6은 예약하고, 7과 8은 고정밀 온도 감지에 사용하고, 마지막 9바이트는 노이즈 방지 CRC 코드를 저장합니다.

DS18B20을 Arduino에 연결하기

DS18B20은 디지털 센서입니다. 디지털 센서는 측정된 온도 값을 특정 바이너리 코드 형태로 전송하며, 이 코드는 arduino의 디지털 또는 아날로그 핀에 입력된 다음 디코딩됩니다. 코드는 매우 다를 수 있습니다. ds18b20은 1-Wire 데이터 프로토콜에서 작동합니다. 우리는 이 디지털 프로토콜의 세부 사항에 대해 설명하지 않을 것입니다. **필요한 최소한의**상호작용의 원리를 이해한다.

1-Wire에서 정보 교환은 다음 작업으로 인해 발생합니다.

> 초기화 - 측정 및 기타 작업이 시작되는 신호 시퀀스의 정의. 마스터 장치는 리셋 펄스를 보낸 후 센서가 작동을 수행

할 준비가 되었음을 나타내는 존재 펄스를 제공해야 합니다.

데이터 쓰기 - 데이터 바이트가 센서로 전송되고 있습니다.

데이터 읽기 - 센서에서 바이트를 수신합니다.

센서로 작업하려면 소프트웨어가 필요합니다.

아두이노 IDE

OneWire 라이브러리, 버스에서 여러 센서를 사용하는 경우 DallasTemperature 라이브러리를 사용할 수 있습니다. OneWire 위에서 실행됩니다.

장비에서 다음이 필요합니다.

하나 이상의 DS18B20 센서;

아두이노 마이크로컨트롤러;

커넥터;

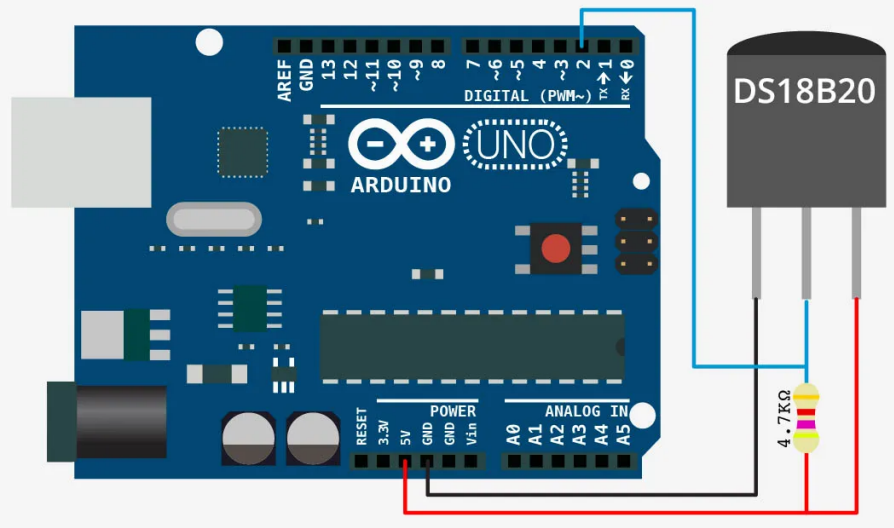
4.7kOhm 저항(하나의 센서가 연결된 경우 4~10K 값의 저항이 사용됨);

회로 기판;

컴퓨터에 연결하기 위한 USB 케이블입니다.

센서는 Arduino UNO 보드에 간단히 연결됩니다. 온도 센서의 GND는 Arduino GND에 연결되고, Vdd는 5V에 연결되고, 데이터는 모든 디지털 핀에 연결됩니다.

DS18B20 디지털 센서의 가장 간단한 연결 다이어그램이 그림에 나와 있습니다.



스케치의 온도에 대한 정보를 얻기 위한 알고리즘은 다음 단계로 구성됩니다.

센서의 주소를 결정하고 연결을 확인합니다.

온도를 읽고 측정값을 레지스터에 입력하라는 명령이 센서로 전송됩니다. 절차는 나머지보다 오래 걸리며 약 750ms가 걸립니다.

레지스터에서 정보를 읽고 수신된 값을 "포트 모니터"로 보내라는 명령이 주어집니다.

필요한 경우 섭씨/화씨로 변환됩니다.

DS18B20에 대한 간단한 스케치 예

디지털 센서로 작업하기 위한 가장 간단한 스케치는 다음과 같습니다. (스케치에서 우리는 OneWire 라이브러리를 사용하는데, 이에 대해서는 조금 후에 더 자세히 이야기할 것입니다).

#포함 /* * 디지털 센서 ds18b20와의 상호 작용에 대한 설명 * 핀 8을 통해 ds18b20을 arduino에 연결 */ OneWire ds(8); // 센서와 함께 작동하는 데 사용되는 1-Wire 버스용 OneWire 객체 생성 void setup()(Serial.begin(9600);) void loop()(// DS18b20 센서 바이트에서 온도 결정 data; // 온도 값의 위치 ds.reset(); // 이전의 모든 명령과 매개변수를 재설정하여 상호 작용 시작 ds.write(0xCC); // DS18b20 센서가 주소 조회를 건너뛰도록 명령합니다. 이 경우에만 one device ds.write(0x44) ; // DS18b20 센서에 온도 측정 명령을 내립니다. 아직 온도 값 자체를 수신하지 않습니다. 센서는 내부 메모리 delay(1000)에 입력합니다. // 마이크로 회로 ds.reset(); // 이제 측정된 온도 값을 받을 준비를 하고 있습니다. ds.write(0xCC); ds.write(0xBE); // 값을 보내주세요. 온도 값이 있는 레지스터 // 응답 데이터 가져오기 및 읽기 = ds.read(); // 온도 값의 하위 바이트 읽기 데이터 = ds.read(); // 그리고 이제 더 오래된 것 // 최종 값 예: // - 먼저 값을 "접착"한 다음 // - 해상도에 해당하는 인수를 곱합니다(12비트의 경우 기본값은 0.0625) float 온도 = ((data<< 8) | data) * 0.0625; // Выводим полученное значение температуры в монитор порта Serial.println(temperature); }

지연 없이 ds18b20 센서로 작업하기 위한 스케치

스케치의 속도 저하를 없애기 위해 ds18b20용 프로그램을 약간 복잡하게 만들 수 있습니다.

#포함 원와이어 ds(8); // OneWire 객체 int 온도 = 0; // DS18B20 센서의 온도 값을 저장하기 위한 전역 변수 long lastUpdateTime = 0; // 센서에서 마지막으로 읽은 시간을 저장할 변수 const int TEMP_UPDATE_TIME = 1000; // 검사 빈도 결정 void setup()(Serial.begin(9600);) void loop()(detectTemperature(); // DS18b20

센서에서 온도 결정 Serial.println(temperature); // 수신된 값 인쇄 온도 값 // T (온도 변수가 int 유형이므로 소수 부분은 단순히 폐기됩니다.) int detectTemperature()(byte data; ds.reset(); ds.write(0xCC); ds.write(0x44) ; if (millis() - lastUpdateTime > TEMP_UPDATE_TIME) (lastUpdateTime = millis(); ds.reset(); ds.write(0xCC); ds.write(0xBE); 데이터 = ds.read(); 데이터 = ds.read(); // 값 생성 온도 = (데이터<< 8) + data; temperature = temperature >> 4; } }

DallasTemperature 라이브러리 및 DS18b20

스케치에서 1-Wire를 통한 ds18b20 센서 작업의 일부 측면을 단순화하는 DallasTemperature 라이브러리를 사용할 수 있습니다. 스케치 예:

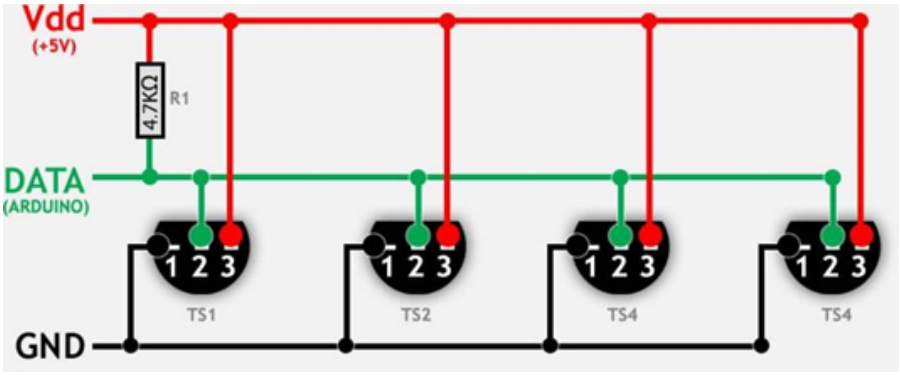
#포함 // 센서가 연결된 Arduino 핀 번호 #define PIN_DS18B20 8 // 객체 생성 OneWire OneWire oneWire(PIN_DS18B20); // 센서 작업을 위한 DallasTemperature 개체를 생성하고 1-Wire 작업을 위한 개체에 대한 참조를 전달합니다. DallasTemperature dallasSensors(&oneWire); // 디바이스 주소를 저장하기 위한 특수 객체 DeviceAddress sensorAddress; void loop(void)(// 온도 센서 측정 요청 Serial.print("Measuring temperature..."); dallasSensors.requestTemperatures(); // ds18b20에 데이터 수집 요청 Serial.println("Done"); // 저장된 온도 값 가져오기 요청 printTemperature(sensorAddress); // 화면에서 무언가를 파싱할 수 있도록 지연 delay(1000);) // 장치의 온도 값을 인쇄하는 보조 함수 void printTemperature(DeviceAddress deviceAddress)(float tempC = dallasSensors.getTempC(deviceAddress); Serial.print("Temp C: "); Serial.println(tempC);) // 센서 주소를 표시하는 도우미 함수 ds18b20 void printAddress(DeviceAddress deviceAddress)(for (uint8_t i = 0) ; 나< 8; i++) { if (deviceAddress[i] < 16) Serial.print("0"); Serial.print(deviceAddress[i], HEX); } }

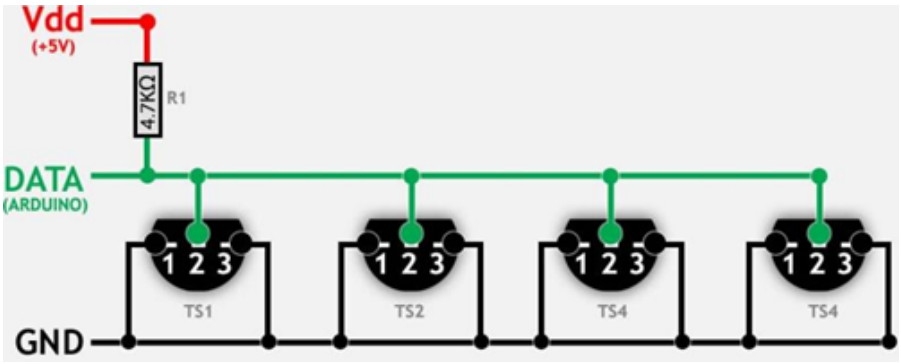
DS18B20 작업을 위한 OneWire 라이브러리

DS18B20은 1-Wire 프로토콜을 사용하여 이미 우수한 라이브러리가 작성된 arduino와 정보를 교환합니다. 모든 기능을 수동으로 구현하지 않도록 사용할 수 있고 사용해야 합니다. . 라이브러리를 설치하려면 아카이브를 다운로드하고 Arduino 디렉토리의 라이브러리 폴더에 압축을 풉니다. 라이브러리는 #include 명령을 사용하여 포함됩니다.

모든 DS18B20 센서는 병렬로 연결되며 하나의 저항으로 모든 센서에 충분합니다. OneWire 라이브러리를 사용하면 모든 센서의 모든 데이터를 동시에 읽을 수 있습니다. 연결된 센서의 수가 10개 이상인 경우 저항이 1.6kOhm 이하인 저항을 선택해야 합니다. 또한, 보다 정확한 온도 측정을 위해서는 Arduino 보드의 데이터 출력과 각 센서의 데이터 사이에 100 ... 120 Ohm 저항을 추가로 넣어야 합니다. 프로그램 실행의 결과로 발행될 고유한 64비트 직렬 코드를 사용하여 특정 값을 얻은 센서를 찾을 수 있습니다.

일반 모드에서 온도 센서를 연결하려면 그림과 같은 회로를 사용해야 합니다.





결론

Dallas DS18B20 칩은 매우 **흥미로운 장치**. 이를 기반으로 하는 온도 센서 및 온도계는 대부분의 작업에 허용되는 특성과 고급 기능을 가지고 있으며 비교적 저렴합니다. DS18B20 센서는 액체의 온도를 측정하기 위한 방습 장치로 특히 인기를 얻었습니다.

추가 기능을 사용하려면 센서 작업의 상대적 복잡성으로 비용을 지불해야 합니다. DS18B20을 연결하려면 약 5K 정격의 저항이 확실히 필요합니다. Arduino 스케치에서 센서로 작업하려면 추가 라이브러리를 설치하고 작업할 특정 기술을 습득해야 합니다. 그러나 기성품 모듈을 구입할 수 있으며 대부분의 경우 스케치의 경우 이 기사에서 제공하는 간단한 예제로 충분합니다.

DS18B20- 댈러스의 디지털 온도 센서. 하나의 디지털 핀과 1-Wire라는 특수 프로토콜만 사용하여 온도 데이터를 전송합니다. 하나의

핀에 여러 센서를 연결할 수 있습니다. 센서는 섭씨 온도를 측정합니다.

사양 DS18B20

센서는 3~5.5V의 전압으로 전원을 공급할 수 있습니다.

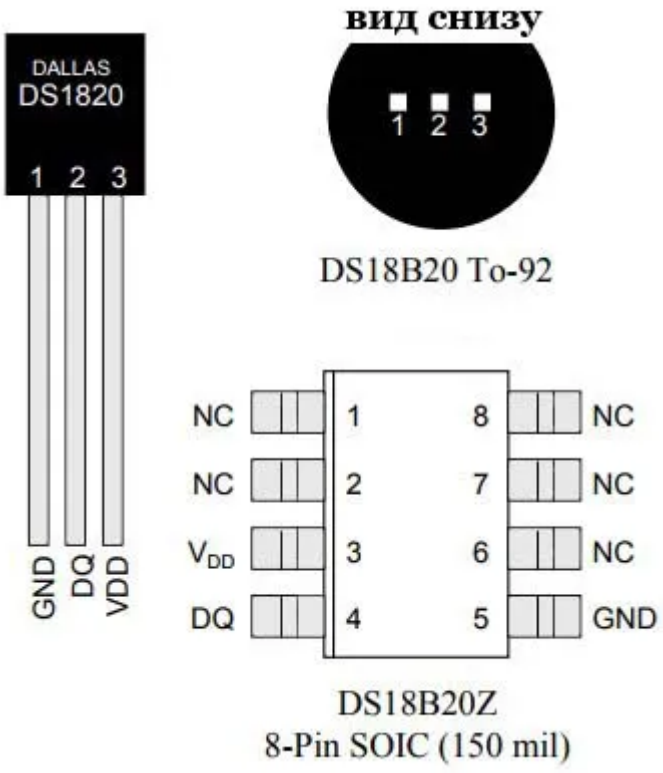
센서는 -55 ~ 125°C의 온도를 측정할 수 있습니다.

센서의 디지털 해상도는 9~12비트입니다.

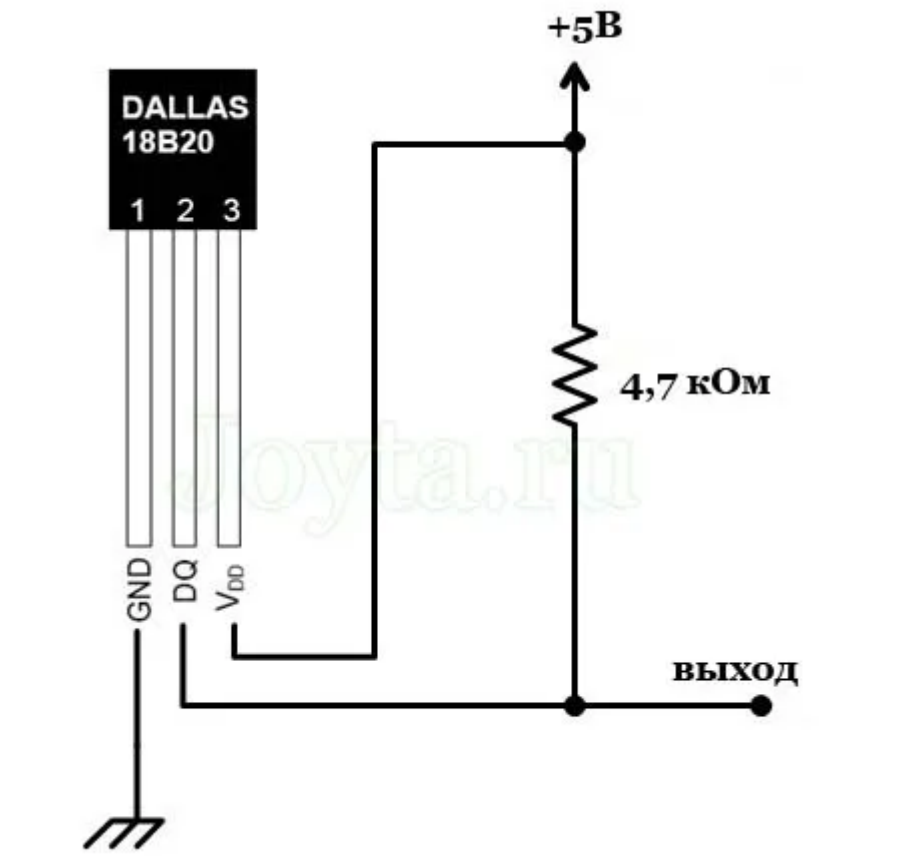
-10 ~ 85°C 범위에서 측정 정확도 +/- 0.5°C

측정 정확도: -55 ~ 125°C 범위에서 +/- 2°C

측정 드리프트 +/- 0.2 °C



배선도 DS18B20



허가란 무엇입니까?

입력 **기술 사양** DS18B20 센서는 다양한 분해능으로 온도를 측정할 수 있다고 보고되고 있다. 해상도는 눈금자와 같습니다. 센티미터 사이의 밀리미터입니다. 유사하게, DS18B20의 해상도에서 이것은 섭씨 온도의 연속 단계 사이의 단계입니다.

해상도는 비트 수를 사용하여 선택됩니다. 선택 범위는 9~12비트입니다. 허가의 선택은 특정한 결과를 수반합니다. 분해능이 높을수록 측정 결과를 기다려야 하는 시간이 길어집니다.

9비트 해상도의 경우 연속 레벨 사이에 2단계가 있습니다.

0.0°C

0.5°C

10비트 해상도의 경우 연속 레벨 사이에 4단계가 있습니다.

0.0°C

0.25°C

0.5°C

0.75°C

이 경우 0.25 °C의 분해능으로 온도를 읽습니다. 10비트 분해능의 측정 시간은 187.5ms로 초당 5.3회 측정이 가능합니다.

11비트 해상도의 경우 연속 레벨 사이에 8단계가 있습니다.

0.0°C

0.125°C

0.25°C

0.375°C

0.5°C

0.625°C

0.75°C

0.875°C

즉, 분해능은 0.125 °C입니다. 11비트 분해능의 측정 시간은 375ms 입니다. 이를 통해 초당 2.6회 측정이 가능합니다.

12비트 해상도의 경우 연속 레벨 사이에는 16단계가 있습니다.

0.0°C

0.0625°C

0.125°C

0.1875°C

0.25°C

0.3125°C

0.375°C

0.4375°C

0.5°C

0.5625°C

0.625°C

0.6875°C

0.75°C

0.8125°C

0.875°C

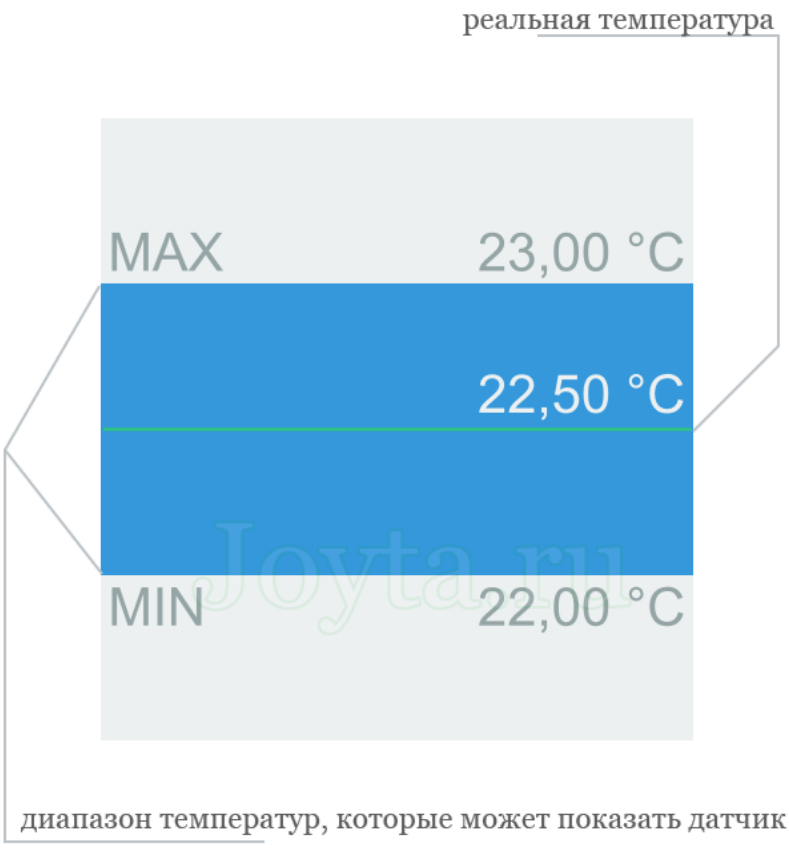
0.9375°C

따라서 분해능은 0.0625 °C입니다. 12비트 분해능의 측정 시간은 약 750ms입니다. 즉, 초당 1.3회 측정할 수 있습니다.

측정 정확도란 무엇입니까?

세상에, 특히 전자 제품에서 완벽한 것은 없습니다. 점점 더 많은 돈과 노력을 들여야만 완벽에 가까워질 수 있습니다. 이 센서도 마찬가지입니다. 당신이 알아야 할 몇 가지 부정확한 내용이 있습니다.

기술 사양에 따르면 -10 ~ 85°C의 측정 범위에서 DS18B20 센서의 정확도는 +/- 0.5°C입니다. 이는 실내 온도가 22.5°C일 때 센서가 22~23°C의 측정 결과를 반환할 수 있음을 의미합니다. 즉, 0.5℃ 이상 또는 이하를 나타낼 수 있다. 그것은 모두 센서의 개별 특성에 달려 있습니다.

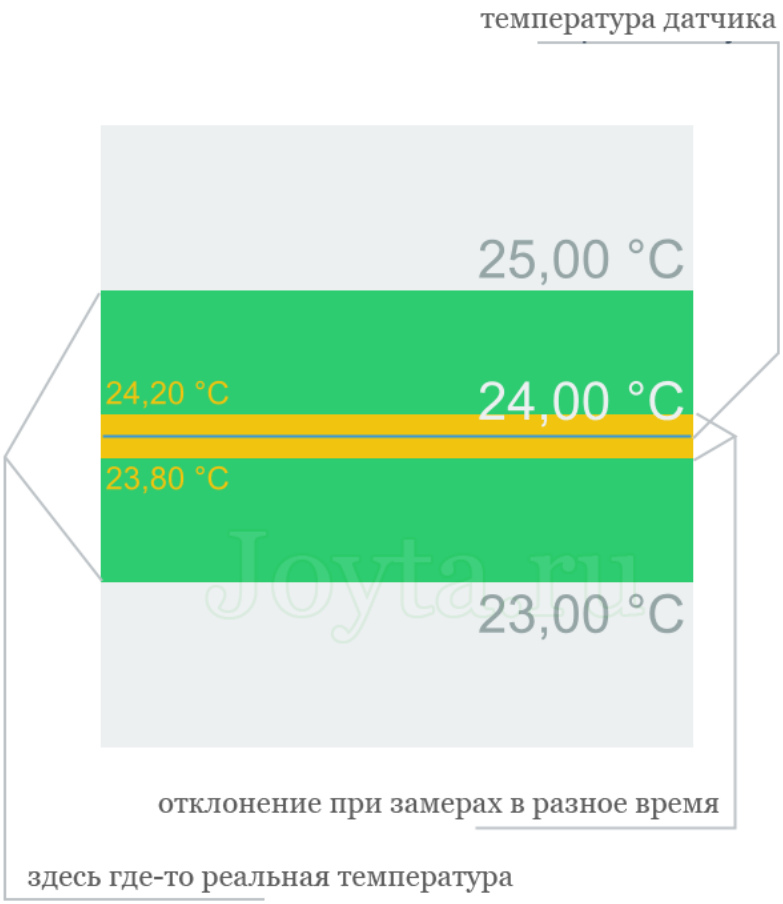


-55 ~ 125°C 범위에서 측정 오류는 최대 +/- 2°C까지 증가할 수 있습니다. 즉, 100 ° C의 온도로 무언가를 측정하면 센서가 98 ~ 102 ° C의 온도를 나타낼 수 있습니다.

이러한 모든 편차는 온도마다 약간 다를 수 있지만 동일한 온도를 측정할 때 편차는 항상 동일합니다.

측정 드리프트란 무엇입니까?

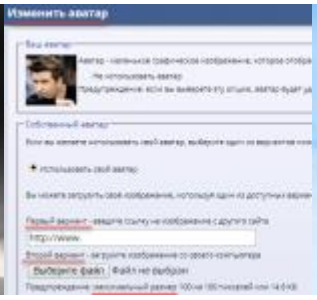
측정 드리프트는 부정확성의 최악의 형태입니다. 측정 드리프트의 본질은 일정한 온도를 측정할 때 한 측정에서 센서가 하나의 온도를 표시할 수 있고 다음에는 다른 온도를 표시할 수 있다는 사실에 있습니다(드리프트 양으로).



온도 센서 드리프트 DS18B20 +/- 0.2 °C. 예를 들어, 실내 온도가 24°C로 일정할 때 센서는 23.8°C에서 24.2°C 사이의 값을 표시할 수 있습니다.

(379.0Kb, 다운로드: 913)

유사한 게시물



생년월일별 사람의 성격

당신의 아바타는 무엇을 말합니까

내가 하품을 한다는 것은 무엇을 의미합니까? 요일 및 시간에 따른 표시

2022년 | 건설 포털 - 마무리. 난방. 통풍. 건축 자재. 설계. 천장