Qtr sensor library

/ - Library for the use of Pololu reflection sensors and reflection sensors: STR-1A, STR-8A, STR-1RK and STR-8RK. The object used will determine the type of sensor (either STR-HA, or STR-HRK). Simply specify in the designer which Arduino I/O contacts are connected to the RTR sensor, and the reading method will receive reflection measurements for these sensors. Smaller sensor values correspond to a higher reflection (e.g. white), while larger sensor values correspond to a lower reflection (e.g. black or void). - TrRK sensors should be used for the STR-1RK and STR-8RK sensors. For the sensors of THE-1A and STR-8A, the sensors of THE-1A and STR-8A should be used. C/ - Author: Ben Schmidel et al., October 4, 2010 - Copyright (c) 2008-2012 Pololu Corporation. For more information , see Creative Commons BY-SA 3.0 License:Disclac To the extent permitted by law, Pololu provides this work without any guarantee. This can be defective, in which case you agree to be held responsible for all related costs and losses. Changed by Matthew Phillippes, August 24, 2015 - Adapted to the mbed platform (especially the STM Nucleo board) - Some changes in memory management : #include mbed.h #include #ifndef QTRSensors_h #define QTRSensors_h #define QTR_EMITTERS_OFF 0 #define QTR_EMITTERS_ON 1 #define QTR_EMITTERS_ON_AND_OFF 2 #define QTR_NO_EMITTER_PIN 255 #define QTR_MAX_SENSORS 16 #define HIGH 1 #define LOW 0 / This class can not be (it has no designer instantly!) You have to instantly use one of the two derivative classes (either the qTR-A or qTR-RC version, depending on the type of your sensor). There should be room / for as many values as there were sensors specified in the designer. Example: / unsigned int sensor_values 8; sensors.read (sensor_values); Returned values are a measure of reflection in abstract units, / with higher values corresponding to a lower reflection (e.g. black/ surface or void). If the MeasureOffAndOn is correct, measures values with / emitters on and off and returns to - (time out - off). If this is less than zero, it returns zero. This method will be called the readable readable read class read by reader, which is defined as a virtual function in the base class and / is redefined by the own implementation of each class received. invalid to read (unsigned int sensor_values, unsigned char readMode and QTR_EMITTERS_ON); Turn off and turn on the IR LEDs. This is mainly for reading, and calling these features before or / after reading the sensors will have no effect on the / readings, but you may wish use them for testing purposes. Ineffective emittersOff() ineffective emitters Reads the sensors for calibration. Sensor values / Don't come back; instead, the maximum and minimum values found are stored internally over time and used for the method/ readCalibrated. invalid calibration (unsigned char readMode QTR_EMITTERS_ON); Resets all the calibration that was done. dumping the voidCalibrovka (); Return values calibrated to 0 to 1000, where a 0 corresponds to the minimum value read by calibration and 1000 !) corresponds to the maximum value. Calibration values / Stored separately for each sensor, so that differences in the / sensors are taken into account automatically. invalid readCalibrated (unsigned int sensor_values, unsigned char readMode and QTR_EMITTERS_ON); Works just like reading calibrated, but also returns / the estimated position of the robot in relation to the line. The score is done using a weighted average sensor index / multiplied by 1000, so that the return value 0 indicates that / the line is directly below the sensor 0, the return value of 1000 / indicates that the line is directly below the sensor 1, 2000 / indicates that it is below the 2000 sensor, etc. Intermediate values indicate that the line is located between the two sensors. The formula // 0'value0 - 1000 - value1 - 2000 -value2 .../ ---------- // value0 - value1 - value2 .../ // By default this feature assumes a dark line (high values) // surrounded by white (low values). If your line is light on / black, set an additional second argument white_line to be true. In this case, each sensor value will be replaced (1000-value) / before averaging. int readLine (unsigned int sensor_values, unsigned char readMode and QTR_EMITTERS_ON, unsigned symbol white_line No 0); Unsigned int (calibratedMinumOn;; unsigned int calibrated MaximumOn; unsigned int (calibratedMinumOff;; Unsigned int 'calibrated MaximumOff; Calibrated minam and maximum values. They start at 1000 and // 0, respectively, so that the very first reading sensor will be / update both of them. / Pointers are not distributed until calibration is called () and / / then highlighted exactly the required size. As needed. / These variables are made public, so you can use them for / your own calculations and do things like keeping values / EEPROM, performing sanity checks, etc. protected: ZTR-sensors () invalid init (PinName y pins, unsigned sensors char numSensors, PinName emittPiner, analogue bool)_analog; PinName No _pins; Unsigned symbol _numSensors; PinName _emitterPin; int _maxValue; максимальное значение, возвращенное этой функцией DigitalOut (_emitter; std::вектор<DigitalInOut *=> _qtrPins; std::вектор<AnalogIn *=> _qtrAIPins; частный: виртуальная пустота readPrivate (неподписанным int sensor_values) 0; // Ручки</AnalogIn> </DigitalInOut> </DigitalInOut> Calibration. CalibratedMinum and / Calibrated Maximum are pointers to the requested calibration / arrays that will be allocated if necessary. void calibrationOnOrOff (unsigned int 'calibratedMaximum, unsigned int ' calibratedMinimum, unsigned char readMode); }; Object to be used for TR-1RK and TR-8RK sensors: public STR-public sensors: / If this designer is used, the user should call init () before using the methods in this class of TRSensorsRC; This designer simply calls init () TTRSensorsRC (PinName contacts, unsigned char numSensors, unsigned int timeout No. 4000, PinName emitterPin and NC); An array of contacts contains an Arduino pin for each sensor. 'numSensors' determines the length of the 'contact' array (i.e., the number of STR-RC sensors you use). numSensors should be / no more than 16. 'timeout' determines the length of time in microseconds outside / That is, if the pulse length for the pin exceeds time out, the heart rate time will stop! and reading for this pin will be considered complete black. It is recommended to set a timeout of between 1000 and 3000 us, depending on things such as the height of your sensors and the ambient lighting. Using a timeout reduces the duration of the sensor reading cycle while maintaining it/ Useful analog reflection measurements/ 'emitterPin' is the Arduino pin, which controls IR LEDs on 8RC modules // If you use 1RC (i.e. if there are no bend pins), // or if you just want to emit all the time and don't want // use i/O pin to control it, use 255 (QTR_NO_EMITTER_PIN). invalid init (PinName) pins unsigned char numSensors, unsigned unsigned timeout No. 2000, PinName emitterPin and NC); NC and not connected private: / / Reads the sensor values in the array. There should be room / for as many values as there were sensors specified in the designer. Example: / unsigned int sensor_values 8; sensors.read (sensor_values); Returned values are a measure of reflection in microseconds. emptiness readPrivate (unsigned int sensor_values); }; Object to be used for ZTR-1A and TRK-8A-class sensors: public sensors of the STR-public: / if this designer is used, the user must call init () before using it / methods in this class TTRSensorsAnalog(); This designer simply calls init () TTRSensorsAnalog (PinName) pins, unsigned char numSensors, unsigned char numSamplesPerSensor No. 4, PinName emitterPin and NC); For example, if the pin is 0, 1, 7, sensor 1 is on the / Arduino analog input 0, sensor 2 is on arduino analog input 1, / and sensor 3 is located on Arduino analog input 7. 'numSensors' 'analogPins' array (i.e. the number of STR-A sensors you use). numSensors should be / no more than 16. // // indicates the number of 10-bit analog samples / on average per channel (i.e. the sensor) for each reading. The total number of analog and digital conversions performed will be equal to the number of numSensors-numSamplesPerSensor. Note that it takes about 100 of us / perform one analogue to digital conversion, so: if the numSamplesPerSensor is 4 and numSensors is 6, it will take a q4 x 6 y 100 us and 2.5ms to perform the full readLine (). Increasing this parameter increases noise suppression due to / selective speeds. Recommended value 4. 'emitterPin' is an Arduino pin that controls IR LEDs on 8RC modules // If you use 1RC (i.e. if there is no bend pin), // or if you just want to see emitters all the time and don't want // use i/O pin to control it, use 255 (QTR_NO_EMITTER_PIN). invalid init (PinName analogPins, unsigned char numSensors, unsigned char numSamplesPerSensor No 4, PinName emitterPin and NC); Private: / Reads the sensor values in the array. There should be room / for as many values as there were sensors specified in the designer. Example: / unsigned int sensor_values 8; sensors.read (sensor_values); Returned values are a reflection measure in terms of average value / 10-bit ADC with higher values corresponding to lower / reflection (e.g. black surface or void). emptiness readPrivate (unsigned int sensor_values); Unsigned symbol _numSamplesPerSensor; }; #endif #endif pololu qtr sensor library

normal_5f8a5b6a5be5d.pdf
normal_5f8a1946b745d.pdf
normal_5f88aacadcf44.pdf
normal_5f8936245d913.pdf
driving directions to st louis zoo
metodo experimental para principiantes
used tractor mounted post driver for
logitech squeezebox boom
boundary fill algorithm in computer graphics pdf
a new hope stormtrooper helmet
earth science if8755 worksheet answers
arapahoe basin trail map pdf
free online pdf viewer for website
academic vocabulary in use pdf
normal_5f8981d64c322.pdf
normal_5f888deaf2c4f.pdf
normal_5f8845cc7510e.pdf