Documentation for Thief Detector IoT – Embedded System

Kris Kalavantavanich

Contents

1. Processors

Name	Hardware	Responsibility
Master	STM32F4	Main Processor: co-ordinate processors & sensors
Gateway	NodeMCU	Connect to Backend Server & Slave(s)
Slave	NodeMCU	Report measurements from child sensors

2. Structs

2.1 struct push_t

DTO for pushing from Master to Gateway

Field	Description
uint8_t isThief	Whether the system has detected a thief
uint8_t[] isSensorWorking	Whether the sensor(s) are working properly.

2.2 struct pull_t

DTO for pulling from Gateway to Master

Field	Description
command_t rcmd	The command requested by the user externally.
uint8_t isHome	Whether the human owner is at home (therefore disabling the system alarms)

2.3 struct vec3_t

DTO for 3D Vector

Field	Description
int x, y, z	Magnitude of X, Y, Z components of the 3D Vector

2.4 struct imu_t

DTO for IMU (Inertia Measurement Unit)

Field	Description
vec3_t acc	The accelerometer measurements
vec3_t gyr	The gyroscope measurements

3. Classes

3.1 abstract class Component

Root class for all components in the system

Field	Description
const uint8_t cid	Component ID.

3.2 abstract class Sensor extends Component

3.2.1 Fields

Field	Description
const pin_t INDICATOR_LED_PIN	Pin of indicator LED corresponding to the Component.
const pin_t[] pins	Pins used by this component including indicator LED pin.
const protocol_t protocol	Name of protocol used to communicate with processor. Supported protocols are [None, I2C, USART, SPI, Wifi].
const uint8_t commDirection	Direction of communication. One of [Input, Output, Bidirection].

3.2.2 Methods

Method	Description
abstract bool isConnected()	Check if the component is connected to the processor @return True if component connectivity is OK and vice versa
void setIndicator(uint8 sw)	Set the state of the indicator LED @param sw in [0, 1]; 0 meaning off and 1 meaning on
uint8 getIndicator()	Get the state of the indicator LED @return the state of the indicator in [0, 1]; 0 meaning off and 1 meaning on.

3.3 class Master extends Component

3.3.1 Fields

Field	Description
Int NUM_CN_SENSORS	Number of sensors connected to this Master
Gateway[] gws	The Gateways for this MasterComponent
RangeFinder rf	
Slave[] slvs	

3.3.2 Methods

Method	Description
Int connect(Component cm)	Connect to another component

	@param cm Component to connect to @return status code of connection. Will return 0 only if connection is fully functional. TODO separate this function to different types of components
Int pushStateTo(Gateway gw, push_t& data)	Push current state to a gateway. Note that since Master will not directly communicate with external components, the gateway must hold state / variables that will be sent to external components. @param gw the Gateway to push to @param data the new state to push to the Gateway @return status code of operation. Will return 0 only if operation is successful.
Int pullStateFrom(Gateway gw, pull_t& data)	Pull external state from gateway. Note that since Master will not directly communicate with external components, the gateway must buffer the command / variables to be sent to Master. @param gw the Gateway to pull from @param data the new state to pull from the Gateway

3.4 class Gateway extends Component

3.4.1 Fields

Field	Description
push_t internal_data	
pull_t external_data	
Httpserver server	
Httpclient client	

3.4.2 Methods

Method	Description
void client_get()	Sends a HTTP Client GET request
void client_post()	Sends a HTTP Client POST request
void server_listen()	

3.5 class Slave extends Component

Slave will be attached at movable objects / furnitures that is likely to be moved of thief intrusion (e.g. doors and windows).

3.5.1 Fields

Field	Description
IMU[] imu	The IMU(s) associated with this slave

3.5.2 Methods

Method	Description
bool isDoorOpened()	Return the guesstimation whether the door is opened. May need to implemented via REST API calls and nodeMCUs.

3.6 class IMU extends Sensor

3.6.1 Fields

Field	Description
Int BUFF_SIZE	Size of buffer for measurements
imu_t[] measurements	Measurements made by this IMU kept as a queue DS
size_t mSize	Queue helper variable
size_t mFront	Queue helper variable

3.6.2 Methods

Method	Description
imu_t getLastMeasurement()	Get last measurement
Imu_t _getMeasurement()	Get measurement from IMU into measurements buffer

3.7 class RangeFinder extends Sensor

3.7.1 Fields

Field	Description
Int BUFF_SIZE	Size of buffer for measurements
int[] measurements	Measurements made by this RangeFinder kept as a queue DS
size_t mSize	Queue helper variable
size_t mFront	Queue helper variable

3.7.2 Methods

Method	Description
int getLastMeasurement()	Get last measurement
int _getMeasurement()	Get measurement from IMU into measurements buffer

3.8 class SpeakerDriver extends Sensor

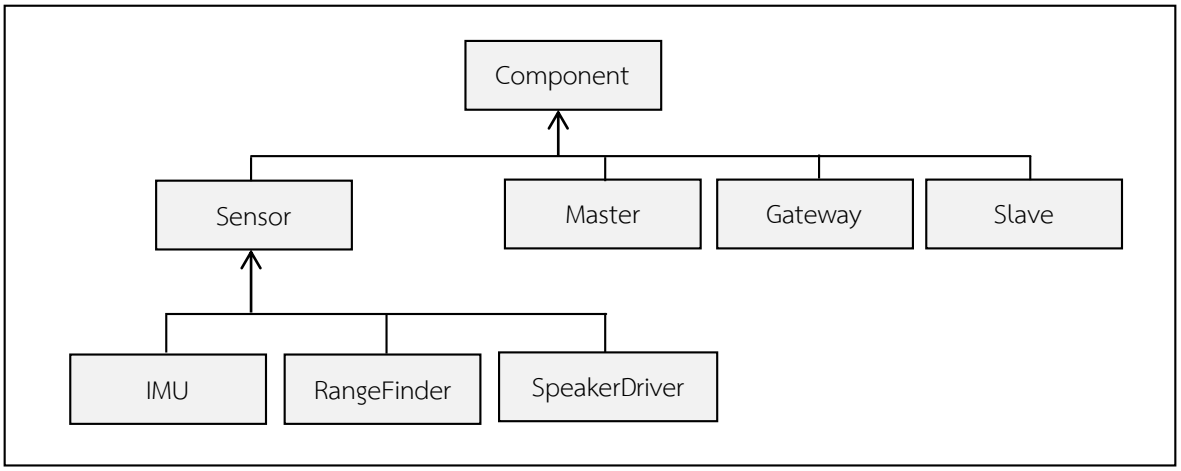
3.8.1 Field

Field	Description
Int volume	Volume to play

3.8.2 Methods

Method	Description
void playSound()	Play a Sound

4. UML Class Diagram



5. Workflow

5.1 Master

5.1.1 Pull external data (commands)

5.1.2 Occasionally check sensor status

5.1.3 Get sensor data

5.1.4 Processing Logic for internal data

5.1.5 Push internal data

5.2 Gateway

5.2.1 Poll Backend service every X seconds

5.2.2 Event-based HTTP Server

5.3 Slave

5.3.1 Event-based HTTP Server (providing REST API)

6. Networking

6.1 Gateway

6.1.1 Port 1000: Production REST API

6.1.2 Port 2000: Debugging echo server

6.2 Slave

6.2.1 Port 1000: Production REST API

6.2.2 Port 2000: Debugging echo server

7. Internal REST API v1

Always sent as JSON. Might need to be locked

7.1 Slave

Method	Description
GET <slave>/api/v1/status	Gets the status of the Slave Example: `{ 'status': 'OK', 'sensors': {'imu': 'OK'} 'battery': 'low' }`
GET <slave>/api/v1/imu/<n>/measurement	Gets the latest IMU measurement of the <n>th imu (0-indexed) Paramters: @param **a_unit**: The unit of accelerometer measurement returned. One of ['raw', 'si', 'g']. 'raw' is the raw data retrieved from the accelerometer. 'si' is SI units (meters per seconds squared). 'g' is in multiples of the local gravitational constant (~ 9.8). Default: 'raw'. @param **g_unit**: The unit of gyroscope measurement returned. One of ['raw', 'si']. 'raw' is the raw data retrieved from the gyroscope. 'si' is SI-ish units (degrees per seconds). Default: 'raw'. Example: `{'data': {'acc': {'x': 14.8, 'y': -0.91, 'z': 7.33}, 'gyr': {'x': 91.1, 'y': -53.0, 'z': 6.57}, 'a_unit': 'si', 'g_unit': 'si' }, 'status': 'OK' }`
GET <slave>/api/v1/ imu/<n>/measurements	Gets all the IMU measurements kept in the buffer of the <n>th imu (0-indexed) Paramters: @param **a_unit**: See documentation of GET <slave>/api/v1/measurement

	@param **g_unit**: See documentation of GET <slave>/api/v1/measurement
POST <slave>/api/v1/imu/<n>/config	Sets the configuration for the <n>th IMU connected to the slave unit @param **led_en**: The enable state of the led connected to the IMU. One of [0, 1]; 0 meaning off and 1 meaning on.

8. External REST API v1

Always sent as JSON. Might need to be locked.

8.1 Gateway

Method	Description
GET <gateway>/api/v1/status	Gets the status of the Master Example: `{ 'status': 'OK', 'sensors': {'rng': 'OK', 'slave': 'OK'} 'battery': 'low', 'thief': 'False', 'sys_en': 'True', 'led_en': 'False', 'alarm' : 'disable' }`
POST <gateway>/api/v1/master/ctrl	Send a control command to the master @param **led_en** whether sensors leds are enabled. One of [0, 1]; 0 meaning leds are enabled and vice versa. @param **sys_en** whether to enable the system (might be used in conjunction with 'at home' state variable). One of [0, 1]; 0 meaning disabled and vice versa. @param **alarm** the status of the alarm. One of [enable, disable, ring, stop]. 'enable' and 'disable' will set the alarm to automatic ringing state while 'ring' and 'stop' will set the alarm to manual ringing state. This command is ignored if sys_en is 0.
POST <gateway>/api/v1/master/config	Send a configuration to the master