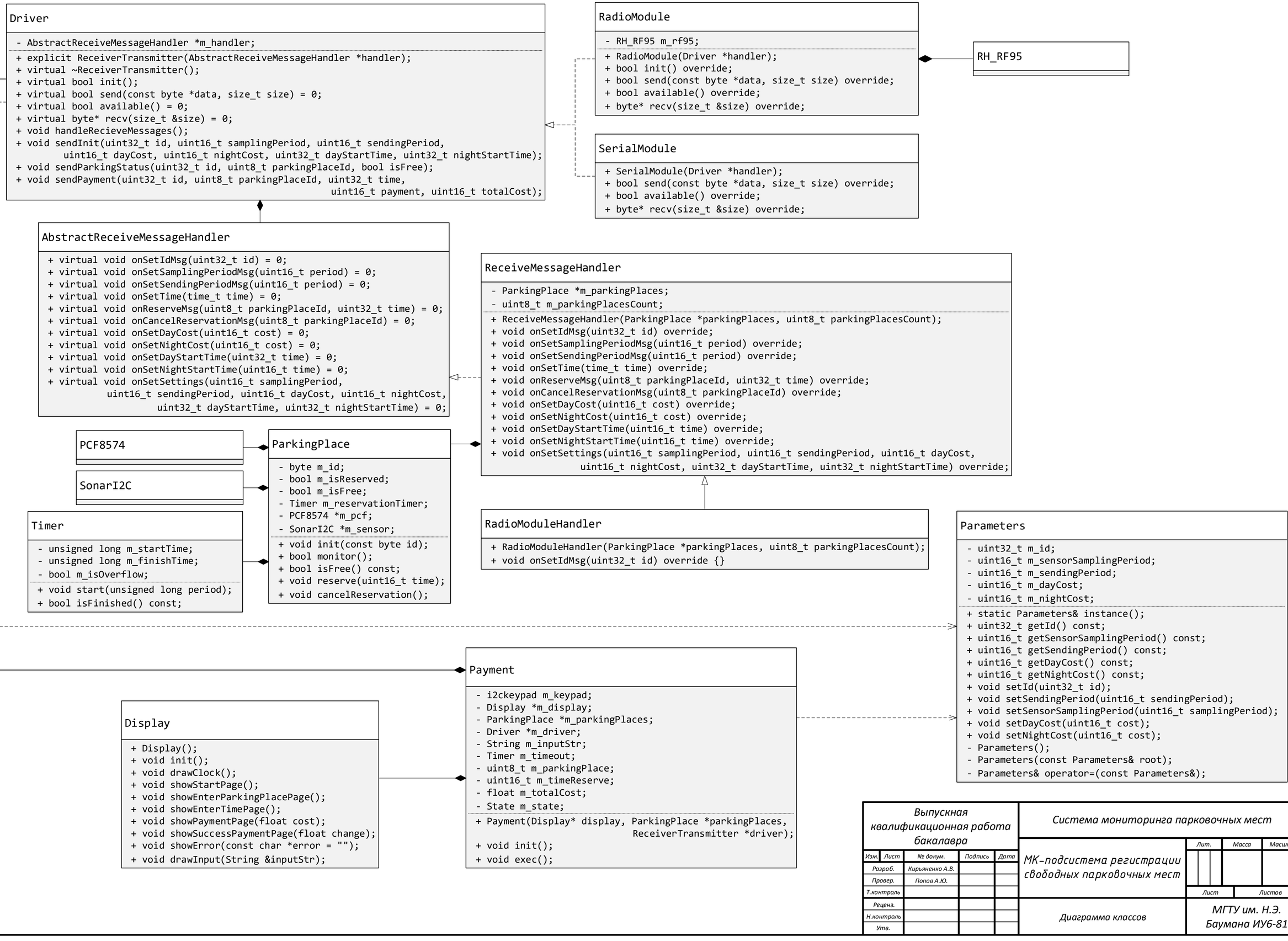
Driver
- AbstractReceiveMessageHandler *m_handler;
+ explicit ReceiverTransmitter(AbstractReceiveMessageHandler *handler);
+ virtual ~ReceiverTransmitter();
+ virtual bool init();
+ virtual bool send(const byte *data, size_t size) = 0;
+ virtual bool available() = 0;
+ virtual byte* recv(size_t &size) = 0;
+ void handleRecieveMessages();
+ void sendInit(uint32_t id, uint16_t samplingPeriod, uint16_t sendingPeriod, uint16_t dayCost, uint16_t nightCost, uint32_t dayStartTime, uint32_t nightStartTime);
+ void sendParkingStatus(uint32_t id, uint8_t parkingPlaceId, bool isFree);
+ void sendPayment(uint32_t id, uint8_t parkingPlaceId, uint32_t time, uint16_t payment, uint16_t totalCost);
RadioModule
- RH_RF95 m_rf95;
+ RadioModule(Driver *handler);
+ bool init() override;
+ bool send(const byte *data, size_t size) override;
+ bool available() override;
+ byte* recv(size_t &size) override;
RH_RF95
SerialModule
+ SerialModule(Driver *handler);
+ bool send(const byte *data, size_t size) override;
+ bool available() override;
+ byte* recv(size_t &size) override;
AbstractReceiveMessageHandler
+ virtual void onSetIdMsg(uint32_t id) = 0;
+ virtual void onSetSamplingPeriodMsg(uint16_t period) = 0;
+ virtual void onSetSendingPeriodMsg(uint16_t period) = 0;
+ virtual void onSetTime(time_t time) = 0;
+ virtual void onReserveMsg(uint8_t parkingPlaceId, uint32_t time) = 0;
+ virtual void onCancelReservationMsg(uint8_t parkingPlaceId) = 0;
+ virtual void onSetDayCost(uint16_t cost) = 0;
+ virtual void onSetNightCost(uint16_t cost) = 0;
+ virtual void onSetDayStartTime(uint32_t time) = 0;
+ virtual void onSetNightStartTime(uint16_t time) = 0;
+ virtual void onSetSettings(uint16_t samplingPeriod, uint16_t sendingPeriod, uint16_t dayCost, uint16_t nightCost, uint32_t dayStartTime, uint32_t nightStartTime) = 0;
ReceiveMessageHandler
- ParkingPlace *m_parkingPlaces;
- uint8_t m_parkingPlacesCount;
+ ReceiveMessageHandler(ParkingPlace *parkingPlaces, uint8_t parkingPlacesCount);
+ void onSetIdMsg(uint32_t id) override;
+ void onSetSamplingPeriodMsg(uint16_t period) override;
+ void onSetSendingPeriodMsg(uint16_t period) override;
+ void onSetTime(time_t time) override;
+ void onReserveMsg(uint8_t parkingPlaceId, uint32_t time) override;
+ void onCancelReservationMsg(uint8_t parkingPlaceId) override;
+ void onSetDayCost(uint16_t cost) override;
+ void onSetNightCost(uint16_t cost) override;
+ void onSetDayStartTime(uint16_t time) override;
+ void onSetNightStartTime(uint16_t time) override;
+ void onSetSettings(uint16_t samplingPeriod, uint16_t sendingPeriod, uint16_t dayCost, uint16_t nightCost, uint32_t dayStartTime, uint32_t nightStartTime) override;
PCF8574
SonarI2C
ParkingPlace
- byte m_id;
- bool m_isReserved;
- bool m_isFree;
- Timer m_reservationTimer;
- PCF8574 *m_pcf;
- SonarI2C *m_sensor;
+ void init(const byte id);
+ bool monitor();
+ bool isFree() const;
+ void reserve(uint16_t time);
+ void cancelReservation();
Timer
- unsigned long m_startTime;
- unsigned long m_finishTime;
- bool m_isOverflow;
+ void start(unsigned long period);
+ bool isFinished() const;
RadioModuleHandler
+ RadioModuleHandler(ParkingPlace *parkingPlaces, uint8_t parkingPlacesCount);
+ void onSetIdMsg(uint32_t id) override {}
Parameters
- uint32_t m_id;
- uint16_t m_sensorSamplingPeriod;
- uint16_t m_sendingPeriod;
- uint16_t m_dayCost;
- uint16_t m_nightCost;
+ static Parameters& instance();
+ uint32_t getId() const;
+ uint16_t getSensorSamplingPeriod() const;
+ uint16_t getSendingPeriod() const;
+ uint16_t getDayCost() const;
+ uint16_t getNightCost() const;
+ void setId(uint32_t id);
+ void setSendingPeriod(uint16_t sendingPeriod);
+ void setSensorSamplingPeriod(uint16_t samplingPeriod);
+ void setDayCost(uint16_t cost);
+ void setNightCost(uint16_t cost);
- Parameters();
- Parameters(const Parameters& root);
- Parameters& operator=(const Parameters&);
Payment
- i2ckeypad m_keypad;
- Display *m_display;
- ParkingPlace *m_parkingPlaces;
- Driver *m_driver;
- String m_inputStr;
- Timer m_timeout;
- uint8_t m_parkingPlace;
- uint16_t m_timeReserve;
- float m_totalCost;
- State m_state;
+ Payment(Display* display, ParkingPlace *parkingPlaces, ReceiverTransmitter *driver);
+ void init();
+ void exec();
Display
+ Display();
+ void init();
+ void drawClock();
+ void showStartPage();
+ void showEnterParkingPlacePage();
+ void showEnterTimePage();
+ void showPaymentPage(float cost);
+ void showSuccessPaymentPage(float change);
+ void showError(const char *error = "");
+ void drawInput(String &inputStr);
Выпускная квалификационная работа бакалавра
Система мониторинга парковочных мест
Изм. Лист № докум. Подпись Дата
Разраб. Кирьяненко А.В.
Провер. Попов А.Ю.
Т.контроль
Реценз.
Н.контроль
Утв.
МК-подсистема регистрации свободных парковочных мест
Лит. Масса Масштаб
Лист Листов
Диаграмма классов
МГТУ им. Н.Э. Баумана ИУ6-81