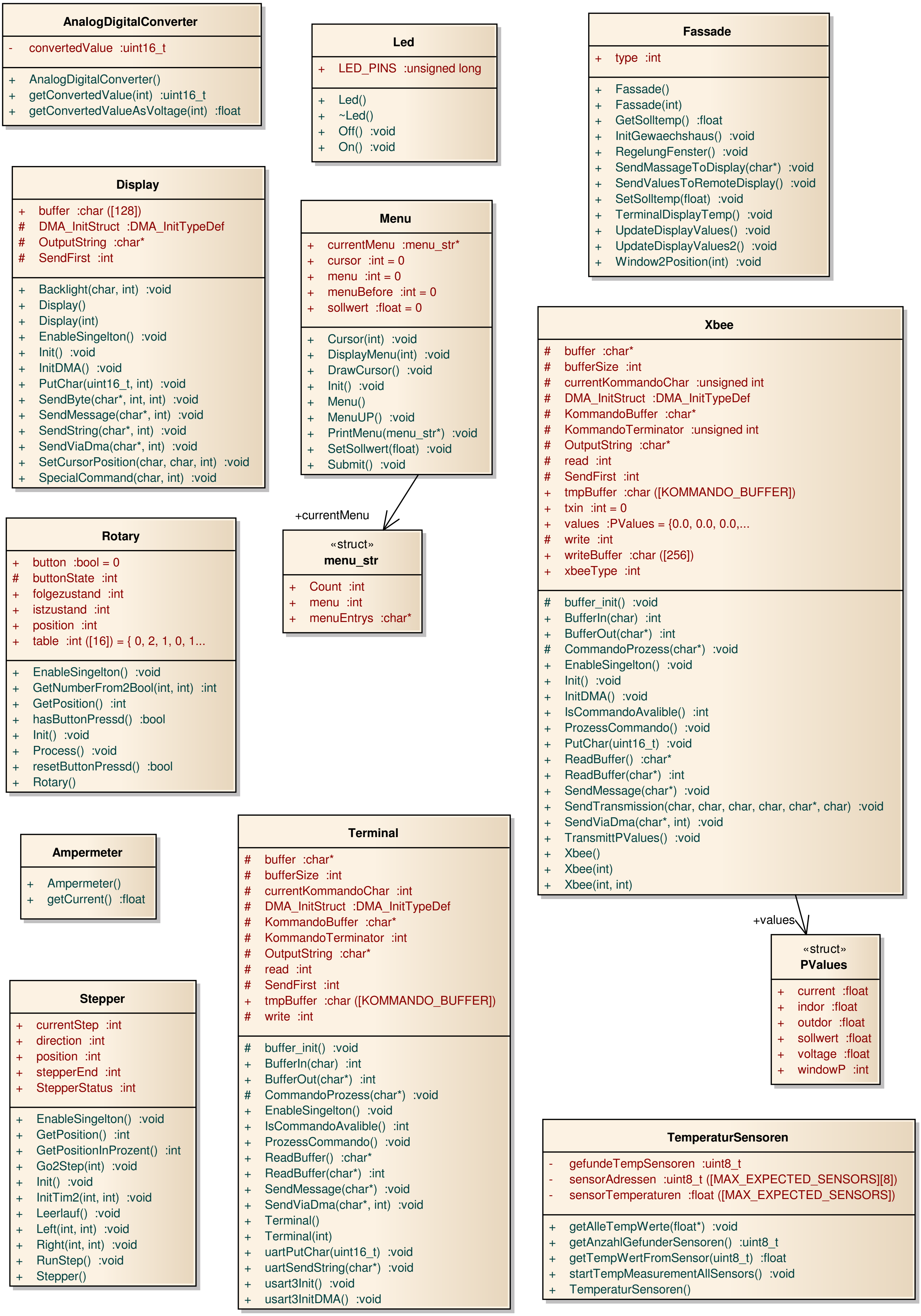

```
AnalogDigitalConverter
-  convertedValue  :uint16_t
+  AnalogDigitalConverter()
+  getConvertedValue(int)  :uint16_t
+  getConvertedValueAsVoltage(int)  :float
```

```
Led
+  LED_PINS  :unsigned long
+  Led()
+  ~Led()
+  Off()  :void
+  On()  :void
```

```
Fassade
+  type  :int
+  Fassade()
+  Fassade(int)
+  GetSolltemp()  :float
+  InitGewaechshaus()  :void
+  RegelungFenster()  :void
+  SendMassageToDisplay(char*)  :void
+  SendValuesToRemoteDisplay()  :void
+  SetSolltemp(float)  :void
+  TerminalDisplayTemp()  :void
+  UpdateDisplayValues()  :void
+  UpdateDisplayValues2()  :void
+  Window2Position(int)  :void
```

```
Display
+  buffer  :char ([128])
#  DMA_InitStruct  :DMA_InitTypeDef
#  OutputString  :char*
#  SendFirst  :int
+  Backlight(char, int)  :void
+  Display()
+  Display(int)
+  EnableSingelton()  :void
+  Init()  :void
+  InitDMA()  :void
+  PutChar(uint16_t, int)  :void
+  SendByte(char*, int, int)  :void
+  SendMessage(char*, int)  :void
+  SendString(char*, int)  :void
+  SendViaDma(char*, int)  :void
+  SetCursorPosition(char, char, int)  :void
+  SpecialCommand(char, int)  :void
```

```
Menu
+  currentMenu  :menu_str*
+  cursor  :int = 0
+  menu  :int = 0
+  menuBefore  :int = 0
+  sollwert  :float = 0
+  Cursor(int)  :void
+  DisplayMenu(int)  :void
+  DrawCursor()  :void
+  Init()  :void
+  Menu()
+  MenuUP()  :void
+  PrintMenu(menu_str*)  :void
+  SetSollwert(float)  :void
+  Submit()  :void
```

```
«struct»
menu_str
+  Count  :int
+  menu  :int
+  menuEntrys  :char*
```

```
Xbee
#  buffer  :char*
#  bufferSize  :int
#  currentKommandoChar  :unsigned int
#  DMA_InitStruct  :DMA_InitTypeDef
#  KommandoBuffer  :char*
#  KommandoTerminator  :unsigned int
#  OutputString  :char*
#  read  :int
#  SendFirst  :int
+  tmpBuffer  :char ([KOMMANDO_BUFFER])
+  txin  :int = 0
+  values  :PValues = {0.0, 0.0, 0.0,...
#  write  :int
+  writeBuffer  :char ([256])
+  xbeeType  :int
#  buffer_init()  :void
+  BufferIn(char)  :int
+  BufferOut(char*)  :int
#  CommandoProzess(char*)  :void
+  EnableSingelton()  :void
+  Init()  :void
+  InitDMA()  :void
+  IsCommandoAvalible()  :int
+  ProzessCommando()  :void
+  PutChar(uint16_t)  :void
+  ReadBuffer()  :char*
+  ReadBuffer(char*)  :int
+  SendMessage(char*)  :void
+  SendTransmission(char, char, char, char, char*, char)  :void
+  SendViaDma(char*, int)  :void
+  TransmittPValues()  :void
+  Xbee()
+  Xbee(int)
+  Xbee(int, int)
```

```
Rotary
+  button  :bool = 0
#  buttonState  :int
+  folgezustand  :int
+  istzustand  :int
+  position  :int
+  table  :int ([16]) = { 0, 2, 1, 0, 1...
+  EnableSingelton()  :void
+  GetNumberFrom2Bool(int, int)  :int
+  GetPosition()  :int
+  hasButtonPressd()  :bool
+  Init()  :void
+  Process()  :void
+  resetButtonPressd()  :bool
+  Rotary()
```

```
Ampermeter
+  Ampermeter()
+  getCurrent()  :float
```

```
Terminal
#  buffer  :char*
#  bufferSize  :int
#  currentKommandoChar  :int
#  DMA_InitStruct  :DMA_InitTypeDef
#  KommandoBuffer  :char*
#  KommandoTerminator  :int
#  OutputString  :char*
#  read  :int
#  SendFirst  :int
+  tmpBuffer  :char ([KOMMANDO_BUFFER])
#  write  :int
#  buffer_init()  :void
+  BufferIn(char)  :int
+  BufferOut(char*)  :int
#  CommandoProzess(char*)  :void
+  EnableSingelton()  :void
+  IsCommandoAvalible()  :int
+  ProzessCommando()  :void
+  ReadBuffer()  :char*
+  ReadBuffer(char*)  :int
+  SendMessage(char*)  :void
+  SendViaDma(char*, int)  :void
+  Terminal()
+  Terminal(int)
+  uartPutChar(uint16_t)  :void
+  uartSendString(char*)  :void
+  usart3Init()  :void
+  usart3InitDMA()  :void
```

```
«struct»
PValues
+  current  :float
+  indor  :float
+  outdor  :float
+  sollwert  :float
+  voltage  :float
+  windowP  :int
```

```
Stepper
+  currentStep  :int
+  direction  :int
+  position  :int
+  stepperEnd  :int
+  StepperStatus  :int
+  EnableSingelton()  :void
+  GetPosition()  :int
+  GetPositionInProzent()  :int
+  Go2Step(int)  :void
+  Init()  :void
+  InitTim2(int, int)  :void
+  Leerlauf()  :void
+  Left(int, int)  :void
+  Right(int, int)  :void
+  RunStep()  :void
+  Stepper()
```

```
TemperaturSensoren
-  gefundeTempSensoren  :uint8_t
-  sensorAdressen  :uint8_t ([MAX_EXPECTED_SENSORS][8])
-  sensorTemperaturen  :float ([MAX_EXPECTED_SENSORS])
+  getAlleTempWerte(float*)  :void
+  getAnzahlGefunderSensoren()  :uint8_t
+  getTempWertFromSensor(uint8_t)  :float
+  startTempMeasurementAllSensors()  :void
+  TemperaturSensoren()
```